Ministère de l'enseignement superieur et de la recherche scientifique
Université A/Mira de Béjaia
Faculté des Sciences Exactes
Département Informatique

Mémoire de Fin de cycle

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

Spécialité : Systèmes Complexes et Technologie de l'Information et de Contrôle (SCTIC)

THÈME

Composition dynamique de services avec QdS dans les environnements ubiquitaires

---

Réalisé par :

M^{lle} OULHACI Tiziri.
M^{lle} TOUDJI Dalal.

Devant le jury composé de :

Président :	D^{r}	OMAR Mawloud.
Rapporteur :	M^{r}	KHANOUCHE M^{ed} Essaid.
Examinatrice :	M^{me}	BOUTRID Samia née AMEZA.
Examinatrice :	M^{lle}	KHOULALENE Nadjet.

JUIN 2012

Remerciements

Nous rendons grâce à notre Dieu, le tout puissant et miséricordieux, pour nous avoir donné le courage et la patience pour mener à bout ce modeste travail.

Nous tenons à exprimer toute notre reconnaissance à notre encadrant Mr KHANOUCHE Mouhamed Essaid pour ses précieux conseils, sa disponibilité, ses encouragements sans lesquels ce travail n'aurait pu voir le jour. Avec patience il nous a guidé tout le long de l'élaboration de ce modeste travail.

Nous adressons aussi nos très sincères remerciements à Dr OMAR Mawloud de nous faire l'honneur d'accepter de présider le jury.

Nous exprimons notre profonde reconnaissance à M^{lle}KHOULALENE Nadjet et M^{me} BOUTRID Samia pour l'intérêt qu'ils ont bien voulu porter à ce travail en acceptant d'en être examinateurs.

Dédicaces

Je dédie ce modeste travail :

à mes très chers parents, à ma grand mére
à mes frères adorés Said, Rachid, Nassim
à toute ma famille de prés ou de loin ,
à tous les étudiants de master2 informatique(promotion2012),
à tous mes amis Sana, Mohand, zazi, Maya, Alaa, Thiwiza
plus particuliérement Hamida, Tiziri, nassima, Sonia et Dahia
Dalal

Je dédie ce modeste travail :

A la mémoire de Mr BAKLI , à mes très chers parents,
à mes adorés frères et sœurs,
à mes nièces et neveux Nihad, Yasmine, Abderahmane, Cherif
à toute ma famille de prés et de loin ,
à tous les étudiants de master2 informatique(promotion2012),
à tous mes amis Dalal, Hamida, Azeddine, Sonia, Nassima, Kahina, Rima, Fouzi, Mouhand, Yazid, ,Souad, Soussou.
Tiziri

Résumé

L'informatique ubiquitaire consiste à offrir aux utilisateurs un accès aux services numériques de leur environnement immédiat. En raison de la prolifération du nombre de services dans les environnements ubiquitaires, le problème de la composition dynamique de services devient une tâche complexe. En effet, lorsqu'un utilisateur requiert un service qui n'est pas directement disponible dans son environnement, le processus de composition combinera à la volée plusieurs services atomiques afin de former un service composite répondant au besoin de l'utilisateur. Ce travail se focalise sur la problématique de la composition automatique et dynamique de services. En effet, nous avons proposé QDSC (Quality-aware Dynamic Service Composition), une approche de composition permettant la satisfaction efficace des besoins fonctionnels et non-fonctionnels (Qualité de Service) sur le plan local (service atomique) et global (service composite) tout en respectant les limites imposées par les contraintes de temps et de disponibilité des ressources tout en optimisant le temps de composition.

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 litérature, ils montrent une réduction en terme du temps d'exécution du plan de composition.

Mots-clés : *Environnements ubiquitaires, architectures orientées services, sélection et composition dynamique de services, qualité de service, service atomique, service composite*.

Abstract

Ubiquitous computing provides users an access to digital services of their immediate environment. Due to the proliferation of the number of services in ubiquitous environments, the problem of dynamic service composition becomes a complex task. Indeed, when a user requests a service that is not directly available in its environment, the composition process will combine on the fly several atomic services in order to form a composite service meeting the need of the user. This work focuses on the problem of automatic and dynamic service composition in ubiquitous environments. We have suggested QDSC (Quality-aware Dynamic Service Composition), a composition approach that allow the effective satisfaction of functional and non-functional (Quality of Service) requirements, respecting the limits imposed by time constraints and resource availability while optimizing the composition time.

To evaluate the performances of the suggested solution, we have conducted simulations in Java. The obtained results were compared with one of the proposed approaches in literature. They show a reduction in terms of concrete plan execution time.

Keywords : *Ubiquitous environments, service oriented architectures, dynamic selection and service composition, quality of service, atomic service, composite service.*

Liste des Abréviations

BPEL	Business Process Execution Language.
CB-sec	Context Based Service Composition.
CORBA	Common Object Request Broker Architecture.
CSD	Concret Service Discovry.
DCOM	Distributed COM.
EU	Environnement Ubiquitaire.
FCoSC	Feasibility and Construction of a Services Composition
IA	Intelligence Artificielle.
PDA	Personal Digital Assistant.
QdS	Qualité de Service.
QoE	Quality of Experience.
QoS	Quality of Service.
RMI	Remote Method Invocation.
SDP	Service Discovery Protocol.
SDT	Services Discovery for a Task.
SMA	Systèmes Multi Agents.
SOA	Service Oriented Architecture.
UML	Unified Modeling Language.
W3C	World Wide Web Consortium.

Table des matières

LISTE DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION GÉNÉRALE

Les avancées technologiques de ces dernières années ont mis l'accent sur la démocratisation des réseaux sans fil et sur la miniaturisation des appareils de communication. Les progrès actuels des systèmes informatiques nous conduisent vers l'ère de l'informatique pervasif, dans laquelle presque tous les objets de notre environnement seront dotés de capacités de calcul et de communication. Cette tendance semble devoir s'accentuer dans le futur, notamment avec les nanotechnologies pour déboucher sur un monde où l'informatique serait omniprésente et invisible, c'est l'informatique ubiquitaire.

L'informatique ubiquitaire représente un pas majeur vers une évolution radicale dans la manière d'interagir avec un système d'information et de communication. Le paradigme de l'informatique ambiante a ouvert des nouvelles perspectives de recherche sur les futurs systèmes d'information. Contrairement à l'approche traditionnelle orientée traitement, l'approche ubiquitaire est orientée service. Elle vise à assister l'utilisateur dans ces activités quotidiennes d'une manière transparente et intuitive. La vision est celle d'un environnement omniprésent, qui assiste l'utilisateur dans ses tâches sans être envahissant, et capable d'exécuter les actions appropriées tout en s'adaptant aux conditions environnementales et aux besoins et préférences de l'utilisateur.

L'informatique diffuse connait cependant des difficultés pour faire communiquer de façon dynamique, spontanée et transparente le nombre croissant de dispositifs informatiques indépendamment de leurs hétérogénéités matérielle et logicielle, à cet effet les intergiciels ont été introduits. La nature particulièrement dynamique de ces environnements, tant au niveau physique que logiciel, a pour conséquence que les informations contextuelles disponibles et les exigences des utilisateurs changent constamment. Par conséquent,la QdS(qualité de service) et le contexte constituent un concept clé dans de tels systèmes et exigent ainsi une bonne compréhension et utilisation.

Ce travail se focalise sur la problématique de la composition automatique et dynamique de services avec QdS dans les environnements ubiquitaires. En effet, la composition consiste à combiner plusieurs services(atomiques) disponibles dans l'environnement de l'utilsateur afin de créer des nouvelles fonctionnalités (services composites) qui ne sont pas directement fournies par les services de base. Or, les enjeux liés à la garantie de la QdS des compositions de services sont aussi importants que les enjeux liés à l'assemblage fonctionnel des services. Par conséquent, la composition de services est un moyen de jonction entre les fonctionnalités élémentaires et les besoins des utilisateurs. Cependant, devant la prolifération du nombre de services disponibles, le problème de combiner plusieurs services pour atteindre un objectif donné, devient une tâche très délicate. La prise en compte

d'informations contextuelles ainsi que le respect des contraintes de QdS imposées par l'utilisateur sur le plan locale et globale est d'une importance capitale dans la réalisation de cette tâche.

Afin de mener à bien notre travail, nous avons scindé ce mémoire en quatre chapitres.

Le premier chapitre présente le contexte général de notre travail. Il s'agit d'abord d'introduire le paradigme de l'informatique ubiquitaire et les principales caractéristiques de son environnement. Nous détaillons ensuite le contexte et la sensibilité au contexte qui ont suscité un grand intérêt des chercheurs.

Le deuxième chapitre est consacré à la composition dynamique de services avec QdS. Nous donnons quelques définitions et concepts liés à la composition de services, puis nous enchainnons par une étude critique et une synthèse des travaux traitant la composition de services avec QdS en environnement ubiquitaires.

Le troisième chapitre est dédié à la proposition d'une approche pour la construction d'un plan de composition d'une façon automatique, en sélectionnant les meilleurs services concrets sensibles au contexte et orientés QdS, et ce dans les deux phases de sélection locale et globale. Nous présentons également dans ce chapitre une étude de cas portant sur des services pour l'assistance et la surveillance des personnes dépendantes.

Le quatrième chapitre quant à lui, porte sur la validation par simulation de la solution proposée et l'évaluation de ses performances en terme de résultats produits et de temps d'exécution.

Enfin, nous concluons ce travail et nous dégageons quelques perspectives.

CHAPITRE 1

PRÉSENTATION DES SYSTÈMES UBIQUITAIRES

1.1 Introduction

Depuis le début des années 90, plusieurs innovations ont marqué le secteur des technologies de l'information, des télécommunications et de la micro-électronique. L'innovation la plus importante est l'émergence de l'informatique ubiquitaire (ubiquitous computing), appelée aussi informatique diffuse (pervasive computing).

L'informatique diffuse se définit comme un environnement saturé de dispositifs informatiques susceptibles de coopérer de façon autonome et transparente afin d'améliorer l'interactivité et/ou l'expérience des utilisateurs [9]. L'objectif est de permettre à ces derniers d'accéder aux différentes fonctionnalités offertes par les divers dispositifs hétérogènes n'importe où et à tout instant, à partir de n'importe quel terminal, tel qu'un téléphone portable, un PDA (Personal Digital Assistant), etc.

Dans ce chapitre, nous dressons un panorama du paradigme de l'informatique ubiquitaire. Nous présenterons notamment ce phénomène, qui y a franchi, depuis quelques années, le seuil de la maturité technique, et a trouvé un réel usage auprès de nombreux utilisateurs. Nous utiliserons des scénarios simples pour mettre en évidence l'un des aspects les plus novateurs de l'informatique ubiquitaire qui est d'assister implicitement et discrètement un utilisateur dans les tâches qu'il accomplit au quotidien. Nous définirons, ensuite, le contexte et la sensibilité au contexte qui ont suscité un grand intérêt des chercheurs. Nous nous intéresserons également aux principes généraux des intergiciels qui assurent l'interopérabilité des applications et qui prennent en compte le contexte de l'utilisateur.

1.2 Informatique ubiquitaire

1.2.1 Présentation de l'informatique ubiquitaire

L'informatique ubiquitaire représente un pas majeur dans une évolution qui a commencé dans les années 1970. Deux autres pas dans cette évolution sont les systèmes distribués et l'informatique mobile. Ces trois pas se focalisent sur des aspects distincts résumés dans la figure 1.1 [2].

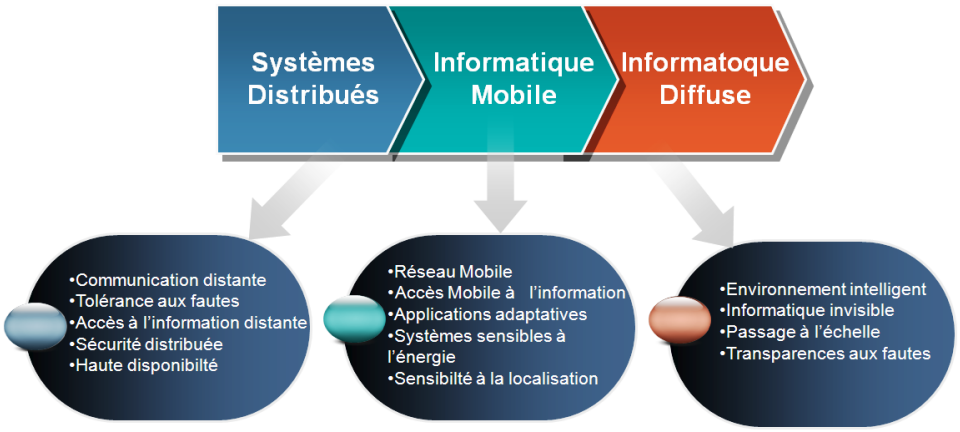


FIGURE 1.1 – Evolution vers l'informatique diffuse.

L'informatique ubiquitaire permet aux utilisateurs de consulter des données n'importe quand, n'importe où, à travers leurs dispositifs d'accès. Le W3C (World Wide Web Consortium)[15] définit l'informatique ubiquitaire, comme un paradigme émergeant de l'informatique personnelle (Personal Computing) [9].

Mark Weiser, pionnier de ce domaine, définit l'informatique ubiquitaire comme une technologie invisible aux utilisateurs, avec lesquels elle entretient des interactions permanentes [4]. Cette nouvelle forme de l'informatique appelée également informatique diffuse, informatique pervasif, intelligence ambiante, informatique omniprésente, ou encore l'internet des objets [1] et dont l'objet viserait à assister implicitement et discrètement un utilisateur dans les tâches qu'il accomplit au quotidien, devenant ainsi la base des systèmes de l'informatique diffuse.

1.2.2 Exemple de scénario

1.2.2.1 Scénario 1

Nous présentons un scénario concret, illustrant l'informatique diffuse, dans lequel un utilisateur Paul interagit avec différents services de son environnement [4].

" Paul " participe à une réunion qu'il doit quitter plus tôt pour se rendre à un rendez-vous au bureau de " Suzanne " situé à l'autre bout de la ville. Dès que Paul quitte la réunion, le flux audio de cette dernière est automatiquement transféré sur son téléphone mobile.

" Paul " prend alors sa voiture, l'ordinateur de bord détecte la conversation téléphonique en cours et la rediffuse immédiatement sur les haut-parleurs du véhicule jusqu'à ce que la conversation se termine. Grâce à un service d'information sur le trafic auquel le véhicule a accès via une infrastructure réseau du réseau routier, un message s'affiche sur l'écran du tableau de bord indiquant à Paul l'existence d'un accident situé à 2 km. Ne pouvant pas se rendre en voiture à son rendez-vous, Paul décide alors de se garer et de poursuivre à pieds.

Ne connaissant pas le quartier, il consulte via son PDA un service d'information local, accessible dans son environnement immédiat grâce au point d'accès Wifi de la ville, qui lui affiche le plan du quartier et l'itinéraire le plus court pour se rendre à son rendez-vous.

Arrivé à temps à ce dernier, Paul sort son ordinateur portable et le connecte au réseau local. Faute de réseau sans fil, Paul n'a pas pu découvrir un service permettant de projeter sur un écran la présentation de son projet. Il termine alors son rendez-vous en synchronisant des documents de son portable avec ceux d'un dispositif de stockage de données appartenant à Suzanne [4].

D'après le scénario, on peut dégager quelques concepts de l'informatique diffuse :

✓ L'aspect dynamique de l'environnement de l'utilisateur : Paul découvre et accède aux services de son environnement, alternativement, à l'aide d'un téléphone mobile, de dispositifs embarqués, d'un PDA ou d'un ordinateur portable, en fonction du contexte dans lequel il se trouve.

✓ Le comportement de l'environnement de l'utilisateur est non déterministe : les interactions initiées par l'utilisateur n'aboutissent pas systématiquement à un même résultat étant donné l'aspect dynamique de son environnement.

1.2.2.2 Scénario 2

Le présent scénario est un scénario de surveillance à domicile. La figure 1.2 donne un aperçu de cette maison intelligente. Chaque pièce est équipée de capteurs adaptés à sa fonction. Par exemple, dans la salle de séjour on trouve des détecteurs de présence, des détecteurs d'incendie, des capteurs de température, et de luminosité. En présence d'une personne les détecteurs de luminosité par exemple, doivent permettre de régler la luminosité de la pièce en fonction de l'éclairage ambiant (la lumière du jour, etc.), baisser automatiquement le chauffage dans les chambres pendant qu'on les aère, fermer les volets roulants aux heures chaudes de la journée plutôt que d'augmenter la climatisation.

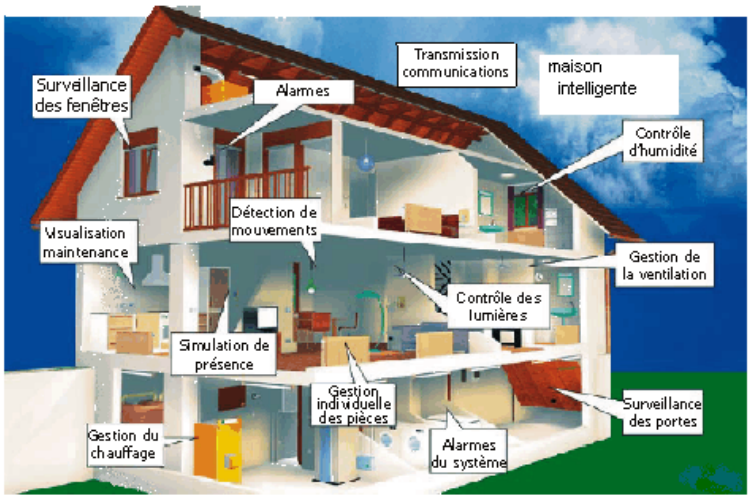


FIGURE 1.2 – Maison intelligente.

1.2.3 Définition d'un environnement ubiquitaire

Grâce aux progrès réalisés ces dernières années dans la miniaturisation de composants électroniques et à l'émergence des technologies de communication sans fil, un nombre croissant d'objets de notre quotidien intègre des dispositifs électroniques grâce auxquels ils deviennent communicants. Ces équipements déployés à travers des réseaux ubiquitaires ont pour objectif d'accompagner l'utilisateur partout et de lui offrir une multitude de services.

Ces objets communicants peuvent être des dispositifs informatiques, des périphériques, ou des équipements de domaines aussi variés que l'électronique grand public, l'électroménager, les télécommunications, la domotique ou l'automobile. Chacun de ces objets dispose de sa propre unité de traitement, puissance de calcul, capacité mémoire, et connectivité réseau [5].

Finalement, l'utilisateur se retrouve au centre d'un espace composé d'objets physiques hétérogènes, dotés de capacités de communication filaire ou non, voir la figure 1.3 [2].

L'environnement numérique de l'utilisateur se définit donc comme un environnement ubiquitaire formé d'une fédération spontanée et dynamique d'objets communicants [4].

1.2.4 Caractéristiques des environnements ubiquitaires

L'informatique ubiquitaire se base sur les systèmes distribués et l'informatique mobile, mais qu'elle y ajoute des caractéristiques comme les espaces intelligents, l'invisibilité, le passage à l'échelle localisé, la prise en compte de l'hétérogénéité des technologies intégrées, etc [9]. Dans ce qui suit, nous détaillons chacune de ces caractéristiques.

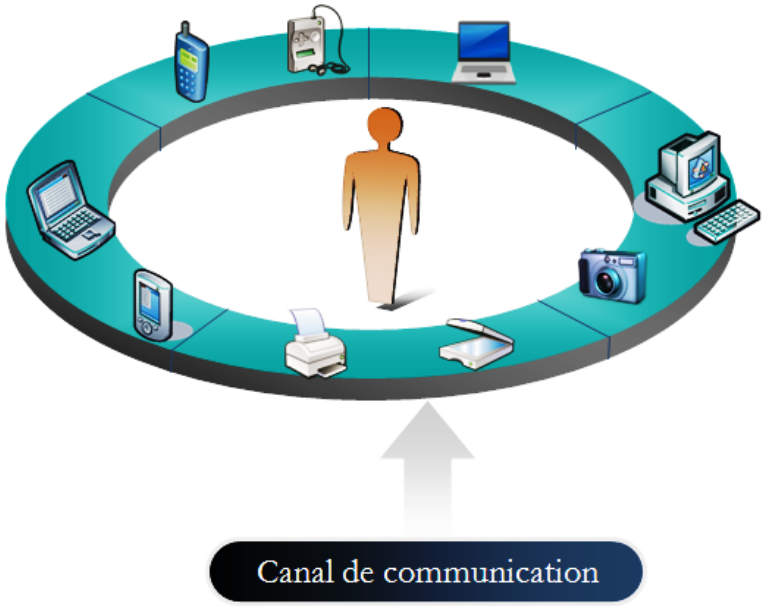


FIGURE 1.3 – Ubiquité des équipements informatiques.

1.2.4.1 Les espaces intelligents

Ils mélangent deux mondes différents à savoir les environnements physiques et les infrastructures de traitement de l'information. Dans les espaces intelligents, ces deux mondes peuvent s'influencer entre eux, à titre d'exemple, l'éclairage d'une pièce peut dépendre du profil d'un utilisateur ; un logiciel embarqué dans un dispositif personnel peut se comporter différemment en fonction de la situation géographique de l'appareil [1].

1.2.4.2 Hétérogénéité

Les services d'un environnement ubiquitaire sont susceptibles d'être déployés sur des objets communicants de différentes sortes (PC, téléphone portable, PDA, etc.). Ces objets sont potentiellement hétérogènes à plusieurs niveaux : matériel, système d'exploitation et langage de programmation. Ces dispositifs doivent interagir entre eux d'une manière transparente malgré les différences entre leurs capacités matériels et logiciels.

1.2.4.3 La dynamicité

L'environnement de l'informatique ubiquitaire est dynamique dans la mesure où les entités qui le composent peuvent l'intégrer ou le quitter à tout moment. L'apparition/disparition de ces objets peut être provoquée par la mobilité de l'utilisateur (par exemple à travers son entrée ou sa sortie d'une zone de couverture particulière.

1.2.4.4 Contrainte des ressources

A propos des dispositifs d'accès propres à l'informatique ubiquitaire, Rahwan et al. [16] soulignent certaines de leurs caractéristiques et contraintes techniques telles que : i) le stockage limité ; ii) la puissance de traitement limitée [9].

Compte tenu de ces caractéristiques, Hristova et al. [17] donnent quelques conseils pour concevoir et développer des applications exécutées sur des dispositifs mobiles : utiliser des logiciels dont l'exécution mobilise peu de ressources mémoire, utiliser des mécanismes minimisant la latence et améliorant la performance de la transmission des données sur des réseaux sans fil, définir un plan d'exécution des applications (et leurs tâches) sur le dispositif mobile, concevoir des interfaces simples, etc. [9].

1.2.4.5 L'invisibilité

L'idée est d'anticiper les besoins de l'utilisateur afin de lui demander explicitement des informations le moins de fois possible. Cela permet aux utilisateurs d'interagir avec les systèmes ubiquitaires au niveau du subconscient ; ils y prêteront attention seulement quand ce sera nécessaire.

1.2.4.6 Le passage à l'échelle localisé

Les études existantes sur le passage à l'échelle ne considèrent pas la localisation des entités ; par exemple dans un serveur web, on essaie de servir le plus de clients possible indépendamment de leur situation géographique. Dans le cas des systèmes ubiquitaires, la plupart des interactions ont lieu entre des entités proches, ce qui peut être exploité pour améliorer le passage à l'échelle. En effet, quand un utilisateur s'éloigne d'un certain environnement ubiquitaire, on pourra réduire le nombre d'interactions qu'il a avec les éléments de cet environnement, car elles seront moins pertinentes [1].

1.2.4.7 Application distribuée

Les environnements ubiquitaires sont de nature distribuée. En effet, les dispositifs qui se trouvent dans un tel environnement se situent dans des endroits différents et proposent leurs propres services. Ces dispositifs sont interconnectés via des réseaux filaires ou sans fil, et exploitent les infrastructures (middlewares) de communication développées dans le cadre des systèmes distribués telles que CORBA, DCOM, etc. Ces infrastructures permettent de présenter les services, les exécuter, et les échanger entre les différents dispositifs si nécessaire [14]

1.2.4.8 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 standard sans fils incluent le Blutooth, Wi-Fi, GPRS, IEEE 802.11a, IEEE 802.11b, IEEE 802.11g), etc. La vitesse lente de

ces technologies par rapport au réseau filaire a toujours été une barrière pour exécuter de grandes applications et déplacer des données volumineuses sur les mobiles [14].

1.3 Quelques domaines d'application

L'informatique ubiquitaire englobe un large éventail de domaines d'application qui touchent pratiquement la quasi-totalité des services de la vie quotidienne. Dans ce qui suit, nous développons les domaines les plus importants.

1.3.1 Le domaine publique

Le service le plus simple pouvant illustré l'impact de l'informatique ubiquitaire est le domaine des transports. Les réseaux sans fils ont rendu possible la communication inter-véhicules dont l'objectif est d'améliorer la sécurité sur les routes ou encore proposer des orientations aux conducteurs.

Dans ce cadre, plusieurs projets ont vu le jour, tel que le projet PLAiiMOB [13]. Le principe de base de ce projet consiste en un échange d'informations entre différents véhicules proches afin, par exemple, d'avertir le conducteur de la présence d'un accident ou d'un obstacle sur la chaussée quelques centaines de mètres plus loin. Les véhicules peuvent ainsi obtenir des informations très variées grâce à leurs voisins [13].

1.3.2 Le domaine médical

L'essence même de la vision de l'informatique ubiquitaire consiste à mettre les nouvelles technologies de l'information et de la communication aux services des utilisateurs notamment dans le domaine médical.
Plusieurs projets ont été réalisés, parmi lesquels " l'assistance médicale pour diabétiques ".En effet l'opérateur français SFR, en partenariat avec l'association française des diabétiques, a lancé le service " T+diabète ". Le principe du service consiste à embarquer sur l'ordinateur de poche ou le PDA du patient une application permettant de lire, de sauvegarder et d'analyser son taux de glycémie. Un taux est mesuré par le malade lui-même via un lecteur connecté au mobile par liaison sans fil Bluetooth [13].

1.4 Les intergiciels pour l'environnement ubiquitaire

1.4.1 Définitions

Parmi les définitions les plus répandues du mot " intergiciel", nous trouvons :

Un intergiciel est une couche logicielle autorisant des logiciels distribués sur différentes stations (ou nœuds) distantes à communiquer de manière transparente, indépendamment des standards réseaux sous-jacents. En d'autre terme, un intergiciel est un logiciel présent sur un nœud du système réparti, qui offre aux composants applicatifs présents sur ce nœud les abstractions d'un modèle de répartition [7].

L'objectif générique d'un intergiciel est de faciliter le développement et l'exécution des applications. En ce qui concerne la distribution des entités d'une application, l'intergiciel masque l'hétérogénéité des réseaux de communication, des ressources matérielles, des systèmes d'exploitation et des langages de programmation.

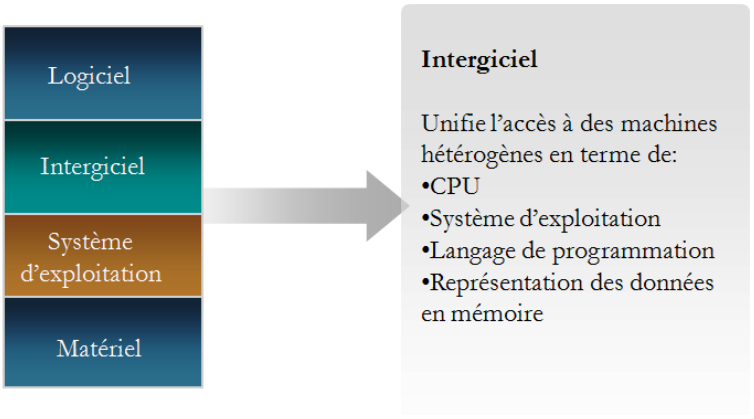


FIGURE 1.4 – Architecture de l'intergiciel.

1.4.2 Rôle d'un intergiciel

L'utilisation d'une couche logicielle intermédiaire s'avère une alternative très prometteuse dans le sens où elle doit [7] :

– Cacher la répartition, c'est-à-dire le fait qu'une application est constituée de partie interconnectées s'exécutant à des emplacements géographiquement répartis ;

– Masquer l'hétérogénéité des composants matériels, des systèmes d'exploitation et des protocoles de communication utilisés par les différents dispositifs de l'environnement ubiquitaire [7] ;

Cette couche intermédiaire de logiciel, schématisée sur la figure 1.5, est désignée par le terme générique d'intergiciel. Un intergiciel peut être à usage général ou dédié à une classe particulière d'applications [7].

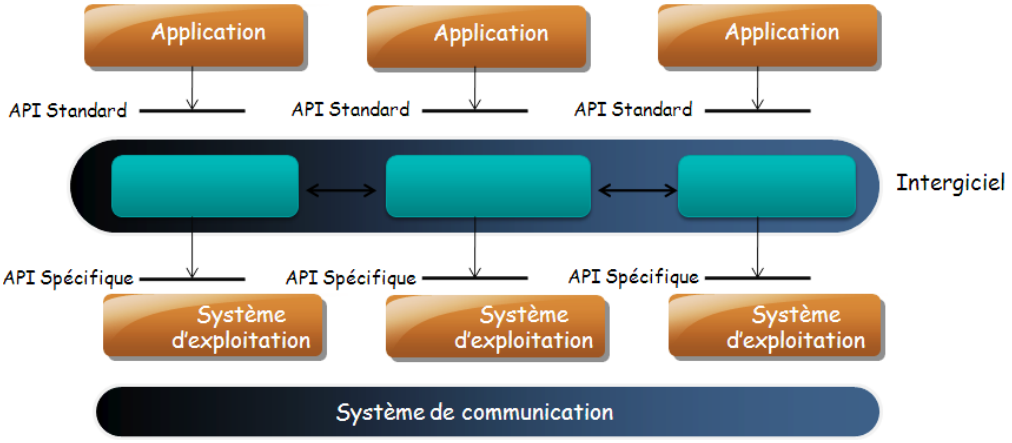


FIGURE 1.5 – Intergiciels de communication.

Il existe une très grande variété d'intergiciels de communication, parmi ces derniers on peut citer, les intergiciels à architecture distribuée qui se basent sur le principe d'appel de procédures à distance, afin de pouvoir invoquer des objets distants de façon transparente. Parmi ces intergiciels CORBA (Common Object Request Broker Architecture), RMI (RemoteMethod Invocation) et DCOM (Distributed COM) , ont permis à certaines entreprises de répondre dans une certaine mesure aux problèmes d'intégration.

Cependant, à cause de l'utilisation des interfaces propriétaires, ces projets se sont avérés souvent coûteux et délicats. L'arrivée depuis quelques années des services Web et surtout leur large succès auprès des grands acteurs de l'informatique donne un second souffle aux SOA [14].

1.5 L'architecture Orientée Service (SOA)

1.5.1 Définition

L'architecture SOA (Service Oriented Architecture) 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 applications [11].

Ce type d'architecture convient le mieux à l'environnement de l'informatique ubiquitaire, qui permet une utilisation homogène des composants logiciels hétérogènes présents dans l'environnement [5].

1.5.2 Principes fondamentaux

Afin de mettre en œuvre une architecture SOA avec succès, il ne suffit pas d'avoir les capacités technologiques. Il convient également d'adopter un certain nombre de principes importants au niveau

de la conception, du développement et de la gestion. Il existe de nombreux principes pouvant être appliqués à une SOA, on peut cependant dénoter quatre principes fondamentaux qu'il est nécessaire de respecter [11].

1.5.2.1 Couplage faible

Le couplage est une métrique indiquant le niveau d'interaction entre deux ou plusieurs composants logiciels. Deux composants sont dits couplés s'ils s'échangent de l'information. On parle de couplage fort (ou serré) si les composants échangent beaucoup d'information et de couplage faible (ou relaxé) dans le cas contraire.

Dans une architecture SOA, le couplage entre les applications clientes et les services est faible. Cela signifie qu'il y a une séparation logique qui isole le client du service afin d'éviter une dépendance physique entre les deux. Pour ce faire, le client communique avec le service par échange de messages dans un format standard. Il est ainsi possible de modifier un service, par exemple pour y ajouter des fonctionnalités, sans briser la compatibilité avec les applications clients existantes. En cas de couplage fort, la possibilité d'évolution des services serait très limitée [11].

1.5.2.2 Interopérabilité

L'objectif de l'architecture orientée service est de permettre aux clients et services de communiquer et de se faire comprendre peu importe la plateforme sur laquelle ils s'exécutent. Cet objectif peut être atteint uniquement si les clients et les services possèdent un moyen standard de communication entre eux [5].

Ceci est notamment réalisé par l'introduction des services Web dans l'architecture SOA.

1.5.2.3 Découverte

La découverte est une étape importante pour permettre la réutilisation des services. Il faut en effet être en mesure de trouver un service afin de savoir qu'il existe et pouvoir en faire usage.

La découverte de service est soit centralisée ou distribuée. Dans le premier cas, la découverte se fait à l'aide d'un annuaire qui centralise les descriptions des services, tandis que dans le second cas, les fournisseurs de services participent à la découverte de services suivant un modèle de découverte de service passif ou actif. La découverte est passive lorsque ce sont les fournisseurs de services qui envoient périodiquement des annonces dans l'environnement des services qu'ils mettent à disposition, tandis qu'elle est active lorsque les clients diffusent des requêtes décrivant les caractéristiques des services dont ils ont besoin [5].

Il est alors nécessaire qu'un service pour SOA soit conçu pour être suffisamment descriptif pour qu'il soit facilement localisable et accessible via un mécanisme de découverte [11].

1.6 Définition d'une architecture de services ubiquitaires

Une architecture de services permet de standardiser l'accès aux ressources et/ou aux fonctionnalités des objets communicants en les représentant sous formes de services. Un service est défini par un contrat (appelé aussi interface) qui est une spécification abstraite de ses fonctionnalités. Ce contrat décrit : (i) ce que le service fournit, (ii) comment y accéder, et (iii) éventuellement, quelles sont ses propriétés non fonctionnelles. L'architecture se compose de consommateurs de services (ou clients) qui interagissent avec des fournisseurs de services, mettant à disposition un ou plusieurs services.

Un client et un service communiquent via des interactions synchrones ou asynchrones suivant le contrat du service, ce contrat est exprimé dans un langage déclaratif indépendant de tout langage de programmation, ce qui permet de s'abstraire de l'implémentation du service. Les consommateurs et les fournisseurs de services ne connaissent rien de leur implémentation respective.

Les clients ne référencient pas directement des instances particulières de services qui leur sont nécessaires mais leur font indirectement référence par l'intermédiaire de la description de leurs caractéristiques. A l'aide de cette dernière, les clients sont en mesure de localiser/découvrir dynamiquement les instances de services disponibles dans leur environnement en interrogeant un service de découverte via un protocole de découverte de services (SDP pour Service Discovery Protocol) [5].

Les interactions entre clients et services se font en plusieurs étapes figure 1.6 [4] :

- Le fournisseur de services publie la description de ses services auprès du service de découverte sur la figure 1.6.
- Le consommateur de services interroge le service de découverte en lui soumettant la description du ou des services requis (étape 1).
- Le service de découverte renvoie le contrat du service et la référence d'une ou plusieurs instances de services correspondant (étape 2).
- Le consommateur de services initie les interactions avec le fournisseur de service suivant les termes du contrat du service (étape3).

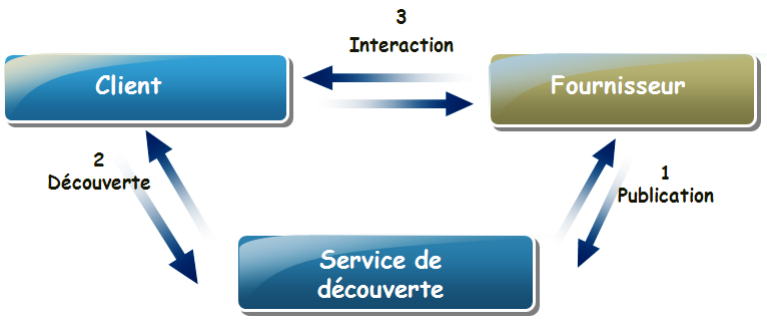


FIGURE 1.6 – Architecture de services.

Une architecture de services ubiquitaires représente un environnement ubiquitaire où les objets communicants se comportent aussi bien comme un consommateur et/ou un fournisseur de service. Au niveau du réseau, les seules parties visibles de leur implémentation sont leurs protocoles de communication respectifs. L'interaction est possible si le comportement du service est connu, c'est-à-dire si son contrat/interface est standardisé et si le consommateur et le fournisseur de services utilisent un même protocole de découverte de services et d'interaction [4, 5].

1.7 La notion de contexte

1.7.1 Définition de contexte

Avec l'apparition de l'informatique ubiquitaire, la notion de contexte a suscité un grand intérêt de la part de nombreux chercheurs.

En effet, l'informatique ubiquitaire a pour rôle de fournir de manière transparente des services à un utilisateur mobile, en prenant en compte, et sans être exhaustifs, différents attributs tels que le temps, la localisation, le profil de l'utilisateur, les ressources disponibles (dispositifs mobiles et accès réseau), etc. Ces attributs constituent ce que l'on appelle classiquement le contexte d'utilisation [6].

Plusieurs auteurs ont essayé de cerner ce concept au moyen de définitions et de classifications, dont nous citons les plus importantes dans cette section.

Schilit et Theimer [18], pour lesquels le contexte se constitué de la localisation de l'utilisateur, des identités ainsi que des états des personnes et des objets qui l'entourent.

Ryan et al [19], pose la définition suivante " le contexte est la localisation de l'utilisateur, son environnement, son identité et le temps.

Pascoe [20] introduit un élément important : l'intérêt. En effet, il définit le contexte comme un sous-ensemble d'états physiques et conceptuels qui ont un certain intérêt pour une entité donnée, Cette notion d'intérêt a été reprise par Abowd, Dey et al. [21] dans leur définition, qui stipulent que : " Le contexte couvre toutes les informations qui peuvent être utilisées pour caractériser la situation d'une entité. Une entité est une personne, un endroit ou un objet que l'on considère pertinent par rapport à l'interaction entre un utilisateur et une application, y compris l'utilisateur et l'application eux-mêmes".

Chen et Kotz [22] : pour lesquels le contexte est l'ensemble des états environnementaux et des paramètres qui déterminent le comportement d'une application ou dans lequel un événement de l'application se déroule et ayant un intérêt pour l'utilisateur ".

Dey [23] a proposé une définition, qui a été adoptée dans la majorité des travaux de recherche en informatique ubiquitaire. Il définit le contexte comme étant toute information pouvant être utilisée pour caractériser la situation d'une entité. Une entité peut être une personne, un lieu, ou tout objet pouvant être considéré comme pertinent dans l'interaction entre l'utilisateur et une application, incluant l'utilisateur et les applications elles-mêmes.

Une analyse faite par (Brezillon et al. 2004) concernant les définitions du terme contexte les a conduits à conclure que la plupart des définitions sont des réponses aux questions suivantes :

1. **qui ?** identité de l'utilisateur courant et d'autres personnes présentes dans l'environnement ;
2. **quoi ?** percevoir et interpréter l'activité de l'usager ;
3. **où ?** localisation de l'usager, ou d'un événement du système ;
4. **quand ?** repère temporel d'une activité, indexation temporelle d'un événement, temps écoulé de la présence d'un sujet à un point donné ;
5. **pourquoi ?** il s'agit de comprendre la raison d'être de l'activité ;
6. **Comment ?** La manière de déroulement de l'activité.

1.7.2 Taxonomie des données contextuelles

Etant donnée la diversité des informations composant le contexte, il est utile de les classer par catégorie pour faciliter leur utilisation. Ces classifications permettent de lister les paramètres que l'on peut prendre en compte dans un système sensible au contexte. Parmi ces classifications on peut citer celle de :

Schilit [18] qui propose de répondre aux questions " où l'on se trouve ", " avec qui " et " quelles sont les ressources à proximité ", et ce afin de caractériser le contexte d'une entité.

Ryan et al [19] proposent les catégories de position, identité, environnement et temps.

Position (localisation) : concerne la position des entités, ainsi que l'orientation, l'altitude et les relations spatiales entre ces derniers.

– **Identité** : à chaque entité on associe un unique identifiant. Une entité peut être une personne, un lieu ou un objet. Un objet est une entité physique ou un composant logiciel.

– **Environnement (état)** : encapsule les caractéristiques des entités. Par exemple, pour un lieu donné, l'état peut caractériser la température, la luminosité, ou le niveau de bruit. Pour une personne, l'état peut se référer à des signes vitaux, sa fatigue ou son activité. Pour les composants logiciels, l'état peut être son temps de réponse, son taux d'utilisation.

– **Temps** : permet d'établir un historique de valeurs permettant d'enrichir le contexte. En effet, l'enchainement et l'ordonnancement d'actions ou d'évènements dans le temps peuvent aussi être importants pour la décision prise par l'application.

Chen et Kotz [1] citent une taxonomie alternative qui divise le contexte en trois catégories principales : le contexte informatique, le contexte utilisateur et le contexte physique.

– Le contexte informatique contient toutes les informations relatives aux ressources matérielles qui permettent l'exécution de l'application.
– Le contexte utilisateur contient des informations relatives à l'identité de l'utilisateur, ses préférences, sa localisation, etc.
– Le contexte physique caractérise l'environnement physique dans lequel se trouvent les personnes et les applications. Cette dernière catégorie inclut également les données temporelles.

1.7.3 Acquisition des informations du contexte

L'acquisition des informations de contexte constitue la première étape dans le processus de traitement du contexte. Concrètement, le contexte fourni à une application peut être acquis à partir des sources diverses. Prenons le cas, d'un utilisateur mobile qui cherche le plus proche restaurant qui répond à ses préférences.

L'utilisateur doit spécifier ses menus préférés et éventuellement d'autres préférences telles que le coût. Le système doit pouvoir répondre aux exigences de l'utilisateur en se basant sur les informations clairement spécifiées par l'utilisateur et sa position actuelle fournie par le système GPS, qui sera transformée en une adresse contenant le pays, la ville, etc.

Cet exemple illustre les différentes sources du contexte. D'abord, les préférences de l'utilisateur sont fournies explicitement par l'utilisateur au système. Ensuite, les coordonnés de l'utilisateur sont capturées par le dispositif de positionnement GPS. Enfin, ces coordonnés sont transformé en adresse. Ainsi, dans l'article [53], Mostéfaoui et al. distinguent trois types de contexte selon les méthodes utilisées pour collecter l'ensemble des informations de ce contexte :

– Le contexte capturé : il s'agit du contexte qui est acquis directement par les capteurs physiques disséminés dans l'environnement telles que les capteurs de température, de bruit, GPS, les caméras,etc.

– Le contexte explicitement fourni : comme nous l'avons cité dans l'exemple précédent, l'utilisateur communique explicitement ses préférences au système (application).

– Le contexte dérivé ou interprété : Il s'agit en fait d'un contexte de haut niveau qui est déduit à partir des informations de contexte de bas niveau (contexte capturé). Dans l'exemple précédent, le pays et la ville où se trouve l'utilisateur représentent un contexte de haut niveau déduit

à partir du contexte de bas niveau qui représente ses coordonnées collectées à partir d'un GPS.

A présent les informations du contexte sont acquises par les dispositifs chargés de cette tâche en l'occurrence les capteurs. Pour exploiter le contexte, une application peut soit accéder directement aux capteurs pour avoir l'information brute, la traiter et puis l'utiliser, ou bien s'appuyer sur un intergiciel qui se charge d'acquérir et stocker les informations de contexte indépendamment de toute application. [14]. (Yachir l'a mis comme un titre à part, le développant d'une manière détaillée).

1.7.4 Sensibilité au contexte

L'informatique ubiquitaire demande aux applications d'être capable de fonctionner dans un environnement extrêmement dynamique en sollicitant le moins possible l'attention des utilisateurs. Ces exigences ont conduit à la conception d'applications dites sensibles au contexte. Ces applications doivent détecter les variations de l'environnement, tels que la localisation et l'identité des utilisateurs et des objets, et adapter leurs comportements en conséquence.

L'idée principale de l'informatique sensible au contexte est de sortir le plus possible les humains de la boucle de contrôle des machines, en réduisant les interactions entre l'humain et la machine. Pour tirer profit des énormes possibilités qu'offrent les informations environnantes sur leurs fonctionnements, les systèmes diffus ont cette caractéristique d'être sensible et de s'adapter au contexte. Ceci nécessite alors une bonne compréhension de ce terme pour son utilisation efficace.

Schilit et Theimer [18] furent les premiers à proposer une définition de la sensibilité au contexte : il s'agit de la capacité d'un système à découvrir et à réagir à des changements dans l'environnement où il se trouve. Ils signalent également l'importance de l'adaptation du système à ces changements.

Abowd, Dey et al. [21] un système est sensible au contexte s'il utilise celui-ci pour fournir à l'utilisateur des informations et des services pertinents. Ces auteurs proposent également une classification des systèmes sensibles au contexte selon leur réponse aux changements de contexte.

1.7.5 Exemple de scénario

1.7.5.1 Scénario 1

L'afficheur d'un téléphone cellulaire augmente sa luminosité lorsqu'il se trouve dans un milieu sombre et le réduit lorsqu'il se trouve dans un milieu éclairé. Dans ce scénario on trouve :

- Service : affichage ;
- La forme (ou qualité) : luminosité augmentée ou réduite ;

– Information du contexte : éclairage dans le milieu où se trouve le cellulaire (le changement de cette information change la qualité de l'affichage.

1.7.5.2 Scénario 2

Un utilisateur a inscrit dans son calendrier d'activités de son téléphone cellulaire une tâche à faire à la date X et à l'heure Y. ces coordonnés temporelles, le cellulaire a été mis par l'utilisateur en mode silencieux parce que l'utilisateur se trouve dans une réunion. Le téléphone cellulaire communique alors avec l'ordinateur portable de l'utilisateur pour voir si ce dernier est en train de l'utiliser, si c'est le cas il lui envoie un message (SMS) qui contient l'activité à faire ; ce message s'affiche directement sur l'écran de l'ordinateur portable. Dans ce scénario on trouve :

– Service : rappel électronique ;
– La forme (ou qualité) : rappel par sonnerie d'un téléphone cellulaire ou par affichage sur l'écran d'un ordinateur portable ;
– Déclenchement d'un service : envoi d'un SMS par le cellulaire à l'ordinateur portable ;
– Informations du contexte : date et heure (le changement de ces valeurs déclenche le service de rappel) et le volume de sonnerie du téléphone cellulaire (si volume est nul alors le service sera fourni sous une autre forme.

1.8 Composition de services

1.8.1 Définition

Un service est dit composé ou composite lorsque son, exécution nécessite d'invoquer plusieurs autres services afin de faire appel à leurs fonctionnalités. La composition de services est une technique permettant d'assembler des services afin d'atteindre un objectif bien particulier, par l'intermédiaire de primitives de contrôle (boucle, test, traitement d'exception, etc.) et d'échange (envoi et réception de messages) [4].

1.8.2 Exemple de scenario

Supposant qu'il existe deux services Web disponibles : findRestaurant, findDirection.

– **findRestaurant** : C'est un service qui prend en entrée un code postal et le repas préféré. En sortie, il fournit le nom, le numéro de téléphone, et l'adresse du plus proche restaurant situé dans la zone du code postal et qui offre le repas préféré qui est déjà spécifié.

– **findDirection** : C'est un service qui prend en entrée une adresse de départ et une adresse de destination. En sortie, il fournit la direction de conduite en montrant sur une carte image, le chemin pour atteindre l'adresse destination partant de l'adresse de départ.

Soit la requête suivante : une personne visite " Alger " pour un voyage d'affaire et il s'arrête à l'hôtel " Sheraton " situé à l'adresse " Staouali, 16001, Alger.". Maintenant, il veut un restaurant traditionnel à proximité de l'hôtel avec la direction de conduite.

Il faut combiner les deux services pour satisfaire conjointement la requête r comme suit : (1) invoquer findRestaurant (" 16001 ", " Traditionnel ") pour avoir le restaurant le plus proche, soit "BarakaRestaurant Boulevard Am. 16001, Alger " et (2) invoquer le service Web findDirection " Staouali, 16001, Alger. ", "BarakaRestaurant Boulevard Am. 16001, Alger " pour avoir la direction de conduite. (Exemple vue en cours).

1.9 Conclusion

Dans ce chapitre nous avons passé en revue les différents aspects de l'informatique ubiquitaire dont l'objectif est de permettre aux utilisateurs de consulter des données n'importe quand, et n'importe où.

L'informatique diffuse connais cependant des difficultés pour faire communiquer de façon dynamique, spontanée et transparente le nombre croissant de dispositifs informatiques indépendamment de leurs hétérogénéités matérielle et logicielle, à cet effet les intergiciels ont été introduits.

Malgré sa grande ferveur auprès des utilisateurs, l'informatique ubiquitaire présente des problèmes dont la composition de services dans les environnements ubiquitaires est l'un des problèmes majeurs.

Plusieurs solutions ont été proposées dans la littérature. Le chapitre suivant sera consacré à la description de quelques solutions du problème précité.

CHAPITRE 2

LES APPROCHES DE COMPOSITION DE SERVICES DANS LES ENVIRONNEMENTS UBIQUITAIRES.

2.1 Introduction

Les environnements pervasifs mettent en œuvre de nombreux objets communicants. Le défi de l'environnement ubiquitaire est l'intégration de services pour l'approvisionnement de nouveaux services personnalisés, plus riches et plus intéressants aussi bien pour des applications, pour d'autres services ou plus communément pour des utilisateurs humains. En effet, si l'utilisateur requière des fonctionnalités, et qu'aucun service n'est seul apte à les fournir, il serait donc nécessaire dans ce cas de découvrir et de composer les services atomiques existants pour répondre à sa requête.

La composition des services est une opération qui se déroule différemment d'un scénario à un autre bien qu'elle soit toujours fortement liée à une étape préalable qui est la découverte des services. Cette dernière présente une phase primordiale pour collecter les services candidats à une composition tout en assurant leur qualité, leur compatibilité et la faisabilité de la composition.

Ce chapitre présente l'état de l'art de la composition de services. Nous introduisons tout d'abord une taxonomie de composition de services qui classifie les deux types d'approches de composition : statique et dynamique. L'objectif ici est d'effectuer une présentation ainsi qu'une étude critique des propositions existantes pour la composition de services avec QdS en environnement ubiquitaires.

2.2 Présentation de la composition de services

2.2.1 Définition

Selon Gardarin [39], " la composition est une technique permettant d'assembler des services afin d'atteindre un objectif particulier par l'intermédiaire de primitives de contrôle (boucle, test, traitement d'exception, etc.) et d'échange (envoi et réception de messages). Les services composants existent au préalable et peuvent ne pas être fournis par la même organisation ".

Selon fabien Baligand [28], la composition de services peut être définie comme étant " le procédé

consistant à combiner des services existants pour former de nouveaux services ".

SelonM. Mrissa [29] : " 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 ". Elle vise à faire interopérer, interagir et coordonner plusieurs services pour la réalisation d'un but [30].

La composition ou l'agrégation de services est une opération qui consiste à construire de nouvelles applications ou services appelés services composites, par assemblage de services déjà existants nommés services atomiques ou élémentaires [58]. La composition spécifie quels services doivent être invoqués, dans quel ordre et sous quelles pré-conditions. Les services basiques peuvent être soient des services atomiques soient des services composites [31].

Un service est dit composé ou composite lorsque son exécution implique des interactions de plusieurs services afin de faire appel à leurs fonctionnalités [54].

2.2.2 Classification des méthodes de composition de services

Les solutions proposées pour la composition de services peuvent être classifiées selon deux axes : lepremierest en fonction du degré de participation de l'utilisateur dans la définition du schéma de composition, et dans ce cas, ces approches peuvent être manuelles, semi-automatiques ou automatiques. Le deuxième repose sur le fait que la sélection et la gestion des services soient faites a priori ou non, dans ce cas l'approche sera dite statique ou dynamique.

Le schéma ci-dessous résume notre classification des différentes méthodes de composition de services.

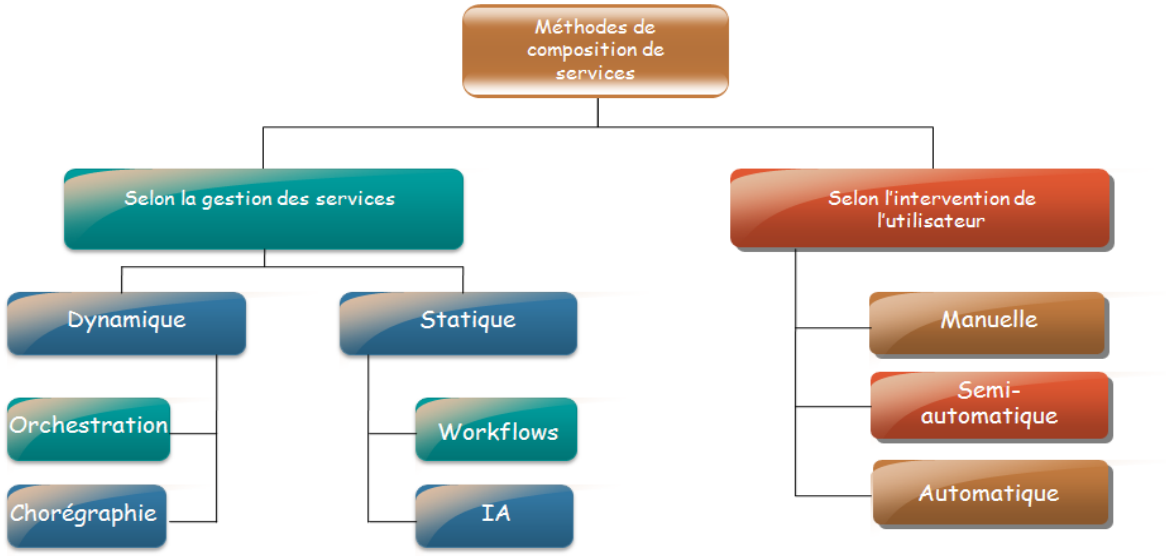


FIGURE 2.1 – Les méthodes de composition de services.

2.2.2.1 Classification selon l'intervention de l'utilisateur

Selon l'intervention de l'utilisateur dans la composition, cette dernière peut être faite de trois manières différentes :

a. **Composition manuelle** : suppose que l'utilisateur génère la composition à la main via un éditeur de texte et sans l'aide d'outils dédiés [32]. La plupart des approches manuelles sont entièrement statiques puisque les services à composer sont préalablement sélectionnés, et le flot de composition est défini a priori [30].

b. **Composition semi-automatique** : les techniques de composition semi-automatiques sont un pas en avant en comparaison avec la composition manuelle, dans la mesure où elles font des suggestions sémantiques pour aider à la sélection des services dans le processus de composition [30].

c. **Composition automatique** : la composition totalement automatisée prend en charge tout le processus de composition et le réalise automatiquement, sans qu'aucune intervention de l'utilisateur ne soit requise [30].

2.2.2.2 Classification selon la sélection et la gestion des services

Selon que la sélection et la gestion des services soient faites a priori ou non, les méthodes de composition de services sont subdivisées en deux catégories, à savoir les méthodes statiques et les méthodes dynamiques.

a) Les méthodes statiques

Dans ce cas, la composition de services est définie a priori, manuellement par le concepteur. Les composants sont choisis et reliés ensemble, avant d'être compilés et déployés. Les techniques de composition statiques sont définies à l'aide de processus métier : orchestration et chorégraphie [31, 33].

L'orchestration

Selon Sanlaville [50],"l'orchestration des services permet de définir l'arrangement et l'enchaînement de ces services selon un canevas bien défini".

L'orchestration décrit la manière par laquelle les services peuvent interagir ensemble tout en incluant l'ordre d'exécution des différentes interactions [31]. Cette technique est réalisée sous le contrôle d'un coordinateur qui gère l'exécution selon un plan défini par le processus métier [33].

L'orchestration exige de définir l'enchaînement des services selon un canevas prédéfini, et de les exécuter selon un script d'orchestration. Le canevas et le script décrivent les interactions entre services en identifiant les messages, et en spécifiant la logique et les séquences d'invocation [34]. Le module exécutant le script d'orchestration est appelé un moteur d'orchestration. Ce moteur (coordinateur) est une entité logicielle qui joue le rôle d'intermédiaire entre les services en les appelants suivant le script d'orchestration [32]. Ce type de composition permet de centraliser l'invocation des services composants.

La figure(2.2) montre le flux d'un workflow (processus métier) dans l'orchestration des services. Un coordinateur prend le contrôle de tous les services impliqués et coordonne l'exécution des différentes opérations des services qui participent dans le processus [34].

La chorégraphie

La chorégraphie de services, contrairement à l'orchestration, ne spécifie pas la composition d'un point de vue central. Elle tente plutôt de définir comment un ensemble de partenaires vont faire collaborer leurs procédés afin d'arriver à un but commun, les procédés de chaque partenaire sont exposés en tant que services [40].Chaque service mêlé dans la chorégraphie connaît exactement quand ses opérations doivent être exécutées et avec qui l'interaction doit avoir lieu [29].

La chorégraphie est une collaboration entre une collection de services, dans laquelle chaque participant au processus d'exécution décrit l'itération qui l'appartient. Ce processus nécessite de spécifier les rôles des services impliqués, les échanges de messages entre les services jouant ces rôles ainsi que les contraintes sur l'ordre d'exécution des messages.

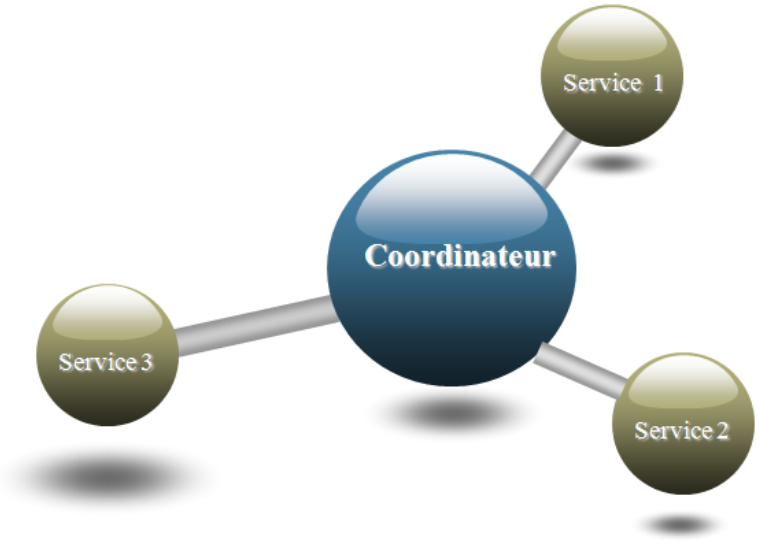


FIGURE 2.2 – Orchestration de services grâce à un coordinateur.

Une hypothèse forte de la chorégraphie de services, est que les participants se sont mis d'accord sur le protocole de collaboration avant de commencer les interactions. Une fois ce protocole établi, chaque participant se comporte de façon autonome tout en respectant le protocole [40].

La collaboration dans la chorégraphie des services peut être représentée comme illustré dans la figure(2.3) :

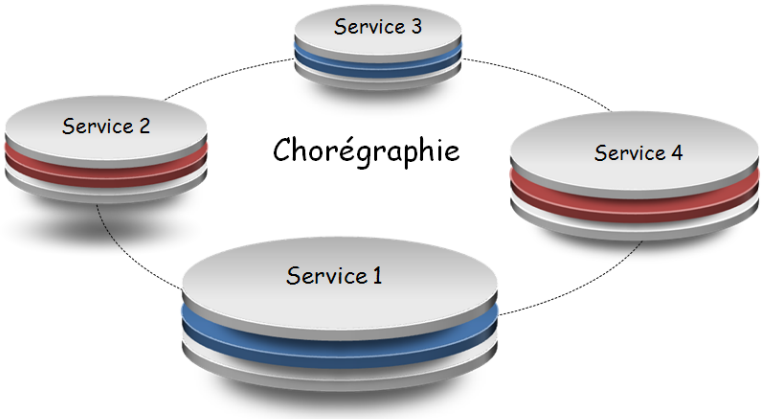


FIGURE 2.3 – Chorégraphie de service.

Depuis la perspective de la composition des services, l'orchestration est un rapprochement plus flexible que la chorégraphie :

✓ Le responsable ou le coordinateur de tout le processus métier est connu ;

✓ Les services peuvent être incorporés sans soucis, parce qu'ils n'ont pas conscience d'appartenir à un processus métier [34].

L'orchestration est considérée comme la meilleure technique de coordination de services puisqu'elle offre plusieurs avantages compétitifs, parmi lesquels nous citons [38] :

- Avantage organisationnel : réduction de l'écart entre la conception et l'implémentation ;
- Avantage de gestion : réduction des coûts de mise en œuvre en proposant l'utilisation de normes et de standards ouverts ;
- Avantages techniques : simplicité et réutilisabilité du code.

b) Les méthodes dynamiques

On appelle composition dynamique, l'agrégation de services permettant de répondre à un objectif précis soumis par un utilisateur en prenant en compte ses préférences. Cette composition peut se faire avant ou pendant l'exécution des services composants [32].

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 [32].

Les approches orientées workflow

Ce type d'approches se base sur le fait qu'un service composite peut être défini 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 [32].

Du côté des compositions dynamiques, le modèle du processus ainsi que la sélection du service sont faits automatiquement [36]. A cet égard, une composition automatique demande des workflows capables de reconnaître les services correspondants à chaque tâche, mais aussi de trouver d'autres services au cas où ceux-ci soient indisponibles ou dévient de leur exécution normale [35].

Approches orientées intelligence artificielle

La plupart des méthodes de composition dynamique se sont inspiré des techniques de l'intelligence artificielles, plus particulièrement les méthodes de planification de l'IA et les théorèmes de preuve par déduction.

Dans ce qui suit, nous présentons quelques approches de composition par planification, par SMA (Système Multi Agents) et par d'autres techniques de l'intelligence artificielle.

1. **Calcul situationnel**

Le calcul de situation [42, 44] est un langage basé sur la logique du premier ordre permettant de représenter et raisonner sur un domaine dynamique. Ce formalisme a été introduit par John McCarthy et Patrick J. Hayes en 1969. Les principaux éléments sont les actions, les fluents et les situations. Le monde est conçu comme un arbre de situations, débutant par la situation initiale S0 et évoluant jusqu'à la nouvelle situation par l'application d'une ou plusieurs actions. Une situation Sdonnée correspond toujours à un historique de l'ensemble d'actions réalisées sur S_0 (voir la figure2.4) [35].

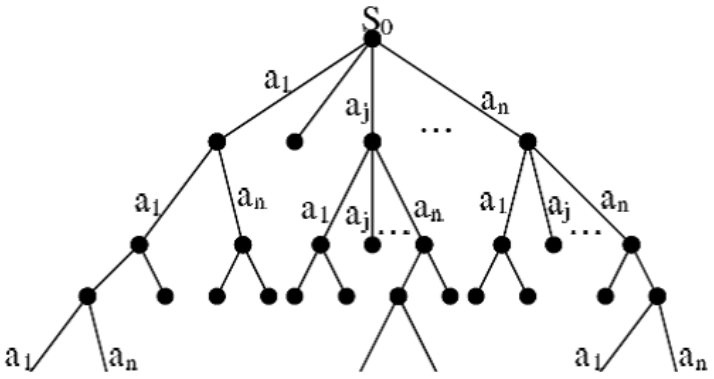


FIGURE 2.4 – Le calcul de situation.

Selon cette technique, 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 [32].

2. **Preuve de théorèmes**
Dans ce type d'approches, les services disponibles et les requêtes utilisateur sont traduites dans un langage du premier ordre. Puis des preuves sont produites à partir d'un prouveur de théorèmes.

3. **Les systèmes multi agents (SMA)**
La composition de services peut être implémentée aussi en utilisant des SMA (Systèmes Multi Agents). Dans ce type d'approches, chaque agent présente un service et sert à satisfaire une partie de la requête de l'utilisateur en utilisant ses propres capacités. Les agents et systèmes multi agents ont beaucoup été étudiés dans la littérature [47, 63].

Jacques Ferber [11] définit un agent comme étant une entité physique ou virtuelle évoluant

dans un environnement dont il n'a qu'une représentation partielle et sur lequel il peut agir. Il est capable de communiquer avec d'autres agents et est doté d'un comportement autonome. Cette définition aborde une notion essentielle : l'autonomie. En effet, ce concept est au centre de la problématique des agents. L'autonomie est la faculté d'avoir ou non le contrôle de son comportement sans l'intervention d'autres agents ou d'êtres humains.

Yves Demazeau [55] définit l'agent comme toute entité réelle ou virtuelle dont le comportement est autonome, évoluant dans un environnement qu'il est capable de percevoir et sur lequel il est capable d'agir, et d'interagir avec les autres agents.

Et enfin Wooldridge [56] définit l'agent comme un système informatique capable d'agir de manière autonome et flexible dans un environnement changeant.

4. **Composition par planification**

La planification en intelligence artificielle consiste à sélectionner et à ordonnancer des actions permettant d'atteindre un but donné à partir d'une base de connaissances sur les actions possibles. Cette dernière contient des préconditions (ce qui doit être vrai avant d'appliquer l'action) et des effets (ce qui arrive après). L'exécution d'un plan modifie les propriétés du monde en le faisant évoluer de l'´état initial jusqu'au but désiré [41, 43, 35].

La figure suivante représente un problème de planification [35] :

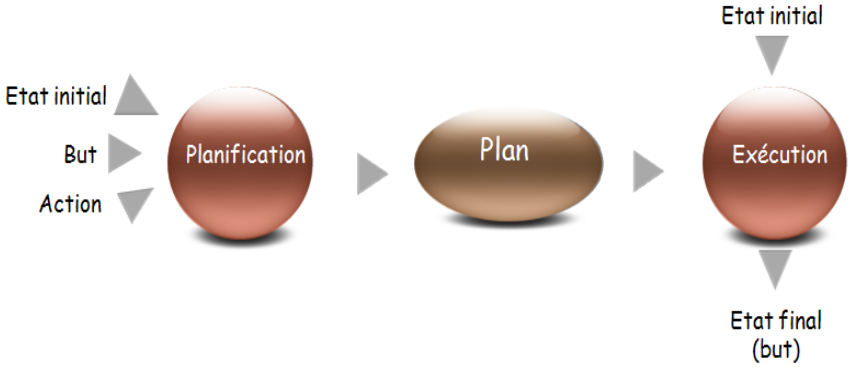


FIGURE 2.5 – Etapes de planification.

2.3 La composition de services dans l'environnement ubiquitaire

2.3.1 Les besoins de l'environnement ubiquitaire

Dans les environnements ubiquitaire, le défi consiste à offrir à l'utilisateur la possibilité d'exécuter sa tâche en composant les services qu'offrent les composants de l'environnement. Les différentes techniques de composition de services décrites plus haut tentent de répondre à la problématique de composition de services. Ces techniques ne prennent néanmoins pas en compte certaines particularités des environnements ubiquitaires. De ce fait, elles ne répondent pas à leurs besoins qui sont, entre autres, l'hétérogénéité, le contexte, les paramètres de QoS et la tolérance aux pannes.

L'hétérogénéité est ce qui caractérise le mieux les composants d'un environnement ubiquitaire. Les approches de composition de services ne prennent en compte l'hétérogénéité que très rarement, puisqu'elles considèrent que les services qui seront intégrés sont développés de sorte à être compatible au niveau de leurs interfaces et leurs protocoles de communication [5].

En ce qui concerne le contexte, l'approche de composition doit implémenter un mécanisme d'adaptation du service offert à l'utilisateur selon les données du contexte. Ce mécanisme doit être en mesure de capturer l'intention de l'utilisateur et voir que le résultat fournie après exécution du processus de composition cadre avec cette intention.

Les paramètres de QoS regroupent la disponibilité, le débit, la fiabilité, temps de réponse,le coût et la sécurité. Ces paramètres doivent être pris en considération lors de la conception du protocole de composition de services pour répondre au mieux à la requête de l'utilisateur et lui offrir un service adapté à ses besoins.

La tolérance aux pannes est importante à prendre en compte lors de la définition d'une approche de composition de services. Le problème est que les environnements ubiquitaires sont très mobiles, rendant ainsi le processus de composition plus complexe à exécuter car aucune certitude n'est donnée quant à la disponibilité du service lors de l'exécution. Un service peut également ne pas fournir les résultats escomptés et les dispositifs hébergeant ces services sont susceptibles de tomber en panne.
Dans ce qui suit, nous présenterons quelques approches de composition relatives à l'environnement ubiquitaire.

2.3.2 Etudes des approches de composition de services

2.3.2.1 Découverte de service et modèle de composition orientée contexte

a) Description

La sensibilité au contexte forme un environnement ubiquitaire capable d'adapter les services aux besoins de l'utilisateur d'une façon implicite. Plusieurs travaux ont été réalisés pour la création des dispositifs et des services plus intelligents, conscients de leurs utilisateurs et environnement.

CB-sec (pour Context Based Service Composition) [37], représente une approche basée sur les systèmes multi-agents et les systèmes sensibles au contexte pour fournir à l'utilisateur les meilleurs services dans un environnement pervasif.Les agents sont des composantes logicielles, capables de fonctionner d'une manière autonome, dans le but de découvrir et de composer des services plus adaptatifs aux besoins de l'utilisateur.

Les types d'informations contextuelles prises en compte dans cette approche sont le contexte de l'utilisateur (rôle, identité, préférences, etc.), le contexte informatique (la connectivité réseau, ressources disponibles, etc.), le contexte temps, le contexte physique (température, vibration...etc.), et le contexte histoire(les informations contextuelles enregistrées dans un laps de temps).
Le Framework adopté dans cette approche repose sur quatre couches principales :

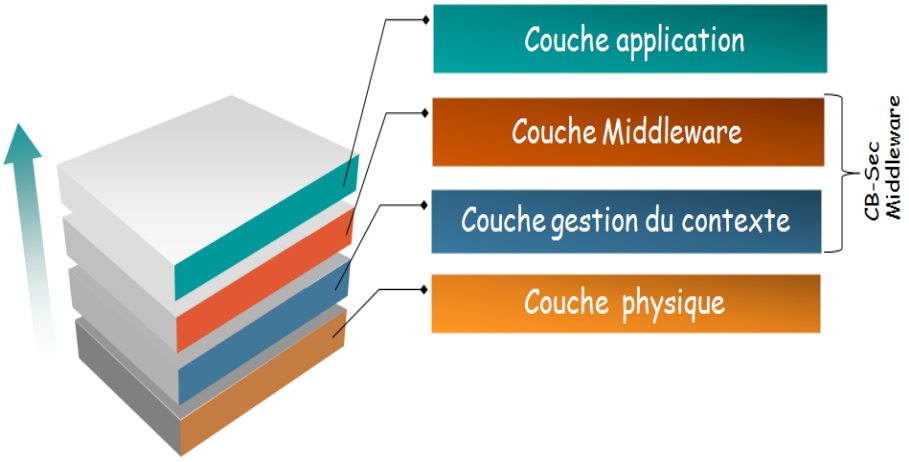


FIGURE 2.6 – Architecture du framework CB-sec

Première couche : la couche d'entités physique, elle représente les entités (dispositifs physiques) appartenant à l'environnement pouvant être impliquées dans le processus d'exécution des tâches.

Deuxième couche : la couche de la gestion du contexte, cette couche est responsable de la collecte d'informations contextuelles, elle est composée de deux modules. Le premier est le moteur de raisonnement ; il s'agit d'un système multi-agent (SMA) où chaque agent est chargé de récolter les

données contextuelles fournies par les capteurs, de les traiter et de les interpréter en informations utiles. Le deuxième est la base de connaissances du contexte ; ce module reçoit les informations contextuelles de chaque utilisateur à partir des agents du moteur de raisonnement, ces informations sont converties en format XML et seront utilisées par le module de découverte.

Troisième couche : la couche middleware, cette couche s'occupe du processus de découverte, de composition et d'exécution de services. Elle est subdivisée en quatre modules : le module de composition, le module de découverte, le module d'exécution, et le module cache.

1. **Le module composition de services** : Ce module est chargé de fournir un processus métier associé à chaque requête complexe de l'utilisateur. Il affine les services sélectionnés par le module de découverte, en utilisant les paramètres de contexte.
2. **Le module de découverte** : Quand un client demande un service, il peut être nécessaire de composer un service complexe à partir des services atomiques existants. Un élément important du module de découverte est l'agent broker. Ce dernier est chargé de trouver les meilleurs services en fonction du contexte et les contraintes fixées par les fournisseurs des services.
3. **Le module d'exécution** : Il réalise l'exécution du service composite dans l'ordre fourni par le module de composition.
4. **Le module de cache** : Le principe du moteur de cache est de stocker des informations pour une réutilisation automatique. La mise en cache peut économiser le temps du traitement et d'accélérer le taux de réponse pour le client.

Quatrième couche : la couche application, elle englobe différentes installations d'interfaces graphiques avec lesquelles les utilisateurs peuvent personnaliser leurs applications en définissant leurs préférences, profil d'autorisation, etc.

b) Bilan

L'approche étudiée est centralisée autour d'un moteur de recherche, ce qui rend le système susceptible de tomber sur un éventuel point de défaillance centrale.

La sélection des services se fait uniquement selon les paramètres contextuels, sans aucune prise en considération de la qualité de service. Dans la conjoncture économique actuelle où le secteur est très concurrentiel, la satisfaction du client est prioritaire, le fait de ne pas adapter l'offre à la demande peut être très pénalisant pour un fournisseur de services.

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.

2.3.2.2 Composition flexible de services d'objets communicants pour la communication ambiante

a) Description

Ce travail [45], s'intéresse à une infrastructure permettant la création d'environnements de communication ambiante grâce à une composition dynamique et flexible des fonctionnalités offertes par des objets communicants divers.

Pour cela, il est nécessaire tout d'abord de faciliter l'intégration de dispositifs hétérogènes, conçus pour des tâches spécifiques, et indépendants d'une infrastructure particulière. D'autre part, un environnement ambiant est par nature instable, et l'infrastructure doit être conçue pour autoriser la dynamicité et l'évolution de cet environnement. Enfin, ces environnements excluent la notion classique d'applications, et imposent de gérer des applications flexibles formées de fonctionnalités assemblées dynamiquement et évoluant en fonction de la situation.

Dans ce travail, les auteurs ont proposé une architecture en couches. Dans la couche inférieure, des objets communicants divers constituent les éléments de base de l'environnement ambiant domestique. Ces objets fournissent des fonctionnalités d'autres éléments du système en tant que services d'objets communicants. La couche intermédiaire est une infrastructure de gestion de services d'objets communicants permettant de publier, de découvrir et d'interagir avec des services hétérogènes. La couche supérieure est un système de gestion flexible des applications, elle assure la création et l'évolution d'applications adaptées aux utilisateurs grâce à la composition dynamique de services. Enfin, un système de gestion de contexte fournit des informations de contexte aux différents éléments du système à partir de données de capteurs (figure 2.7).

FIGURE 2.7 – Architecture générale de la composition flexible de services.

Cette approche porte sur un système multi-agent prenant en compte de manière dynamique, les besoins des utilisateurs ainsi que les informations sur le contexte et les fonctionnalités disponibles afin de créer et d'adapter des applications par composition de services d'objets communicants. Ce système multi-agents servira de médiateur entre les besoins des utilisateurs, les services disponibles

et le contexte.

Un agent assistant se charge de recueillir les besoins courants de l'utilisateur, pouvant lui être fournis directement ou par le biais de règles de déclenchement contextuelles. Pour y répondre, il définit un objectif de composition, qu'il transmet à un agent compositeur. Le fonctionnement du compositeur s'articule autour de cette spécification.

Il en extrait les caractéristiques des différentes fonctionnalités nécessaires, et délègue la recherche des services correspondants à des agents superviseurs. Le rôle d'un agent superviseur est de proposer le service le plus approprié pour réaliser la fonctionnalité dont il est responsable. Lorsque ceux-ci l'ont renseigné, il peut évaluer les compositions possibles et les réaliser en mettant en relation les services appropriés.

Pour cela, l'infrastructure de services fournit à l'agent superviseur une première liste de services dont la description sémantique correspond à la fonctionnalité recherchée. Le superviseur évalue ensuite leur adéquation avec le contexte courant, ainsi que leur interopérabilité avec les services proposés par les autres superviseurs.

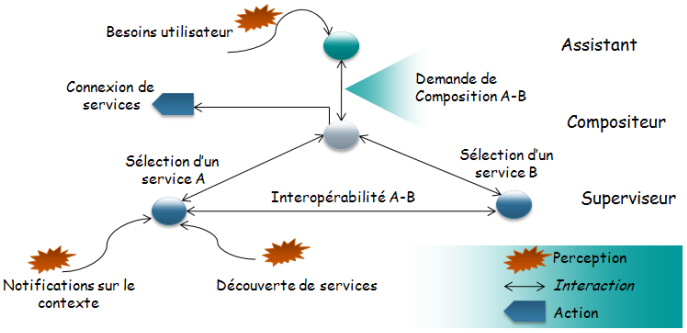


FIGURE 2.8 – Système multi-agents pour la composition de service

b) Bilan

L'approche proposée porte sur un système multi-agents pour la composition flexible d'applications de communication ambiante. Ce système prend en charge l'intégration de dispositifs hétérogènes, leur évolution et la gestion flexible d'applications réparties grâce à l'utilisation de descriptions sémantiques de services d'objets communicants, à la prise en compte des besoins des utilisateurs comme buts et à une gestion réactive du contexte. Par ailleurs, des mécanismes de prise en charge des conflits dans un environnement multi-utilisateurs n'ont pas été pris en considération, ni la QoS (Quality of Service).

2.3.2.3 Approche automatique de composition de services en robotique ubiquitaire

a) Description

Dans ce travail [48], les auteurs ont élaboré une approche permettant de générer automatiquement un plan de composition des services robotiques omniprésents après avoir vérifié sa faisabilité.
L'approche est robuste et transparente pour l'utilisateur. D'abord, l'algorithme de découverte de services SDT (pour Services Discovery for a Task) est exécuté, afin de trouver les services atomiques qui peuvent répondre directement à la requête de l'utilisateur. S'il n'ya pas de services pouvant satisfaire la tâche demandée, l'algorithme FCoSC (pour Feasibility and Construction of a Services Composition) est exécuté afin de trouver un plan de composition.

La robustesse de l'approche provient des quatre règles, définies par les auteurs qui répondent à toutes les relations possibles pouvant exister entre la production de services (l'intersection, l'inclusion, et égalité).

Cependant, les environnements ubiquitaires sont marqués par une grande mobilité, pouvant provoquer quelques perturbations sur les services. Par exemple, le changement de lieux, la panne du serveur, et l'épuisement de la batterie peuvent être à l'origine du changement et de la disparition de services. La solution envisagée à ce problème apparaît sur deux niveaux différents. Si le changement se produit sur un service concret, il suffit de choisir un autre service concret pour le remplacer. En revanche, si le changement se produit sur un service abstrait, il ya échec de tous ses services concrets. Dans un tel cas, le plan sera affecté. Par conséquent, il devient nécessaire de faire une découverte de nouveaux services pour pallier le problème de défaillances.

b) Bilan

Les paramètres de QoS n'ont pas été pris en compte dans cette approche, cela ne reflète pas les réels besoins de l'utilisateur.

De plus, la dynamique de recomposition des services devient impérative lorsque le nombre de défaillances des services abstraits est important. étant donné que chaque service abstrait peut avoir plusieurs services concrets, il est important d'introduire, à ce niveau, une méthode de sélection pour choisir le "meilleur" service concret en termes de probabilité de réponse et d'adaptation aux préférences de l'utilisateur (les paramètres de QdS).

2.3.2.4 Composition automatique de services basée sur des règles

a) Description

Ce travail [49] se concentre sur la composition dynamique de services dans l'environnement ubiquitaire. En effet, les auteurs supposent que :

– **Hypothèse 1**

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 est noté SAi. Il est décrit par un triplet<SA_i^{in} ,SA_i^{out} , SA_i^{cs}, R > tels que :

- SA_i^{in} : Ensemble des entrées de SA_i ;
- SA_i^{out} : Ensemble des sorties de SA_i ;
- R : la réputation du service abstrait ;
- SA_i^{cs} : Ensemble des services concrets de SAi tel que :
  SA_i^{cs}= $cs_{i,1}$, $cs_{i,2}$,..., $cs_{i,n}$ Et $\forall$ $cs_{i,j}$, $cs_{i,k} \in SA_i^{cs}$ / ($cs_{i,j}^{in}$= $cs_{i,k}^{in}$ = SA_i^{in})

$\bigwedge$($cs_{i,j}^{out}$ = $cs_{i,k}^{out}$ = SA_i^{out}).

– **Hypothèse 2**
Le gestionnaire du contexte capture les attributs contextuels à partir du profile utilisateur et du contexte de son environnement et génère ensuite une requête T décrite sous la forme (T^{in}, T^{out}), où (T^{in}, T^{out}) représente les E/S de la requête T.

L'approche de composition de service développée dans cet article est nommée par CDSC (Context-aware Dynamic Service Composition) comprend trois couches :

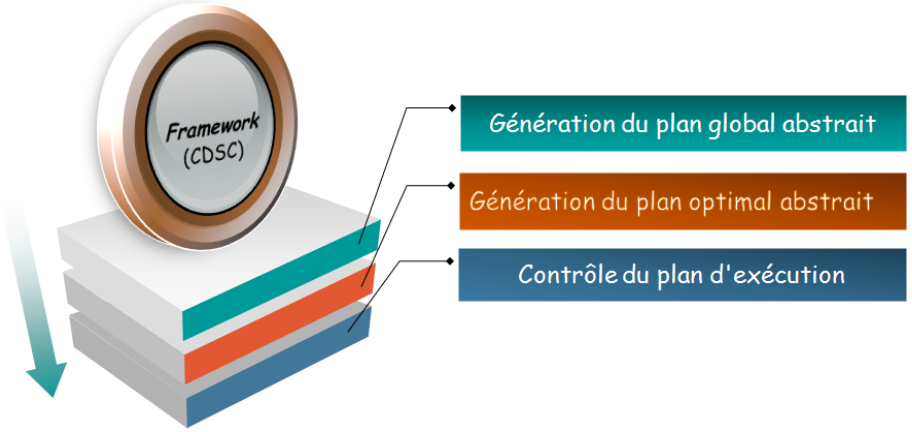


FIGURE 2.9 – Architecture en couche de l'application CDSC.

Première couche : génération du modèle de plan abstrait

Elle consiste à générer automatiquement un plan global abstrait en utilisant un ensemble de règles définis par les auteurs. Le plan contient tous les services abstraits qui pourraient contribuer à composer le service demandé exprimé dans la requête T.

La génération de ce plan peut être considérée comme un problème de construction d'un graphe G. Ce graphe G est une paire (V, E) où V est un ensemble de sommets représentant les paramètres requis, et E est un ensemble d'arêtes représentées par un ensemble de quadruplets (RP_i, RP_j, AS_j, R_j), où RP_i, RP_j V tel 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 de 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 RPi, et qui requiert le paramètre RP_j. Le service abstrait ASi a une réputation RP_i.

Initialement, le graphe G = (V, E) = (RP_0, Φ). RP_0 étant les paramètres de sortie de la requête T.Pour chaque paramètre RPi demandé, des services abstraits seront choisis à partir du répertoire des services abstraits (ASD). Les services sélectionnés, appelés services abstraits candidats, ont au moins un paramètre de sortie appartenant à RPi. Si toutes les données fournies par ces services abstraits candidats incluent les paramètres de la demande RPi, alors tous les paramètres d'entrée des services abstraits candidats seront ajoutés à V et tous les arcs correspondants (RP_i, RP_k, AS_k, R_k) seront ajoutés à E. Dans le cas contraire, le RPi sera supprimé du graphe car aucune solution n'existe.

La génération du graphe G s'arrête lorsque tous les paramètres demandés sont identifiés ou bien lorsque la composition devient impossible.

Deuxième couche : Génération du modèle de plan optimal
Cette couche utilise le plan de la première couche, d'une part pour la génération d'un modèle de plan optimal de composition abstrait et, d'autre part, pour la régénération d'un nouveau plan optimal en cas d'échec.

L'optimisation du plan global est basée sur la réputation et la complémentarité des paramètres de sortie fournis par les services abstraits, afin d'éviter la sélection des services qui offrent les mêmes paramètres. Par conséquent, le plan optimal généré à un nombre de services plus réduit pouvant participer à la composition du service exprimé dans la requête T.

La génération du plan optimal est fondée sur des règles et des techniques de classement définies par les auteurs. L'estimation de la réputation des services abstraits est donnée comme suite :

$$R_{t+1}(a) = \frac{Nb_{success} \pm 1}{Nb_{selection} + 1} \tag{2.1}$$

Où R_{t+1} représente la réputation du nouveau service concret à l'instant t+1.

Le grapheobtenu est un plan inverse du plan de la composition de services, Alors une fois le plan optimal est obtenu, son exécution est faite en traversant le graphe de ses extrémités (ie états finaux) en remontant jusqu'à l'état initial qui représente le paramètre de sortie de la requête de l'utilisateur.

Après la génération du plan optimal, chaque service abstrait sélectionnera le meilleur service concret selon les paramètres de qualité de service QoS.

La troisième couche : Elle a pour fonction la surveillance et le contrôle du plan d'exécution et garantit une adaptation automatique et une tolérance aux défaillances en remplaçant le service concret qui a échoué par un autre qui a 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 en évitant la phase de redécouverte de services et de la régénération d'un nouveau plan global.

Dans le cas où l'exécution du plan de composition a réussi, le service composé sera publié dans le répertoire (CSD) pour une utilisation ultérieure.

b) Bilan

Dans un environnement ubiquitaire, le contexte est riche et varié aussi, il mérite d'être pleinement pris en considération par les dispositifs évaluant la performance des systèmes. Dans cette approche, cela n'a pas été pris en compte dans le processus de sélection et de composition de services.

De plus la QdS n'a pas été clairement développée dans cette approche, c'est à dire, qu'aucune description formelle n'a été proposée pour évaluer la QdS du service concret.

2.3.2.5 Modèle de composition de services basé sur la QdS en robotique ubiquitaire

a) Description

Plusieurs approches se sont penchées sur le problème de composition de services en robotiques ubiquitaires. Cependant la plus part d'entres elles n'intègrent pas les aspects suivants : automatisation du processus de composition des services ; prise en compte de la nature stochastique et dynamique des environnements ubiquitaires ; et intégration de la sensibilité au contexte d'utilisation. C'est dans cette optique que s'inscrivent les travaux de recherche mené dans cet article [51], qui ont conduit à deux contributions majeures :

La première est la proposition d'une nouvelle approche basée sur un raisonnement sur les entrées/sorties des services robotiques ubiquitaires. Cette approche vise à construire un plan pour une composition de services après avoir vérifié sa faisabilité.

La deuxième consiste en la proposition d'une approche de monitoring de la composition de services afin de prendre en compte l'aspect dynamique et stochastique de l'environnement ubiquitaire, elle considère notamment le contexte d'utilisation et la qualité des services disponibles.

Trois mécanismes sont à la base de la méthode proposée, à savoir : l'apprentissage Bayésien, la découverte automatique des services et la recomposition automatique et dynamique du plan de composition en cas de défaillances.

Dans ce qui suit, nous détaillons cette approche qui s'inscrit dans le cadre d'une recherche autour de la composition de services en robotiques ubiquitaires.

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 a déterminé le service le plus approprié parmi l'ensemble des services fournissant la même tâche.

Ce travail se concentre sur la composition horizontale où on explore le contexte des services et leurs probabilités de réponse. L'approche de composition verticale à été développé dans [48], elle est basée sur un algorithme de composition nommé FCoSC (Feasibility and Construction of a Services Composition) et un algorithme de découverte automatique nommé SDT (pour Services Discovery

for a Task).

Probabilité et qualité de service estimée

Les auteurs ont proposé une nouvelle définition pour un service concret en introduisant un **sixième composant** aux cinq déjà proposé dans [48] à savoir la probabilité de réponse. Le but est de prendre en compte la nature stochastique de l'environnement ubiquitaire. Un Service concret CS_i est alors par défini comme suite (CS_i^{in} : inputs, CS_i^{out} : outputs, Prec : pré-conditi, Eff : effects ,QoS : qualité de service et P : probabilité de réponse).

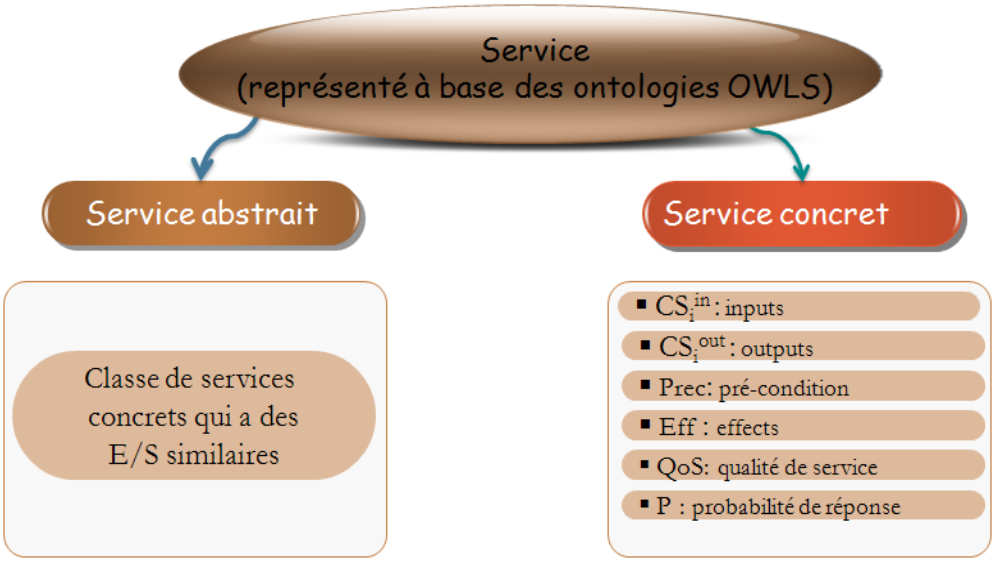


FIGURE 2.10 – Classification des services concrets/abstraits.

1. **Probabilité de réponse**
   un service concret est une action 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 sortie requis, cela signifie qu'après 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 ce framework les auteurs ont utilisé

le modèle d'apprentissage Bayésien.

2. **Apprentissage Bayésien**
   L'algorithme d'apprentissage Bayésien maintient un compteur d'expérience noté exper 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 :

$$P' = ([p * exper] + 1)/(exper + 1), \quad \text{(réponse positive) ;}$$
$$q' = 1 - p', \quad \text{(réponse négative).}$$

3. **Estimation de la qualité de service**
   Chaque paramètre de la qualité de service change en fonction de son contexte, cette différence peut être décrite par des poids associés à chaque paramètre en fonction de l'application désirée et le contexte de l'environnement. Par conséquent, tous les services concrets du même service abstrait peuvent être classifiés selon chaque paramètre de QoS.

Par exemple, si nous somme intéressé par le paramètre " coût ", le service qui a le coût minimum sera classifié en première position(0) et le service qui a le coût maximum sera classifié en dernière position (N-1). Le tableau1 donne la classification des services concrets d'un même service abstrait pour chaque paramètre de la QoS (PQi).

	PQ1	PQ2	...	PQk
Service 1	C11	C12	...	C1k
Service 2	C21	C22	...	C2k
...				
Service N	CN1	CN2		CNk

TABLE 2.1 – Matrice de classification de services concrets

Où
N : nombre de service concrets ;
PQi : paramètre de qualité i ;
P k : est le poids qui reflète le degré d'importance de paramètre PQk pour l'utilisateur et l'application ;
Cik $\in$ [0, N-1] : c'est la classe de service i dans la colonne PQk ;
QoS(i) $\in$ [0,1] : c'est la qualité obtenue une fois que le service i est achevé, il est estimé par la

formule suivante :

$$QoS(i) = \frac{\sum_k P_k * (N - C_{i,k})}{N * \sum_k P_k} \tag{2.2}$$

Après le calcul de la qualité de chaque service satisfait par la formule précédente l'algorithme QoSEA(pour Quality of Service Estimation Algorithm)détermine le meilleur en terme de QoS et de probabilité de réponse. Ce service devient alors l'action pour l'exécuter.

Framework pour le contrôle de processus de composition de services

Le framework proposé est un modèle en trois couches, chacune correspond à l'une des étapes du processus de composition à savoir :
Composition de services abstraits, plan d'exécution, Découverte et recomposition de services abstraits.
Pour réaliser ce framework, les auteurs ont proposé une architecture à 3 couches :

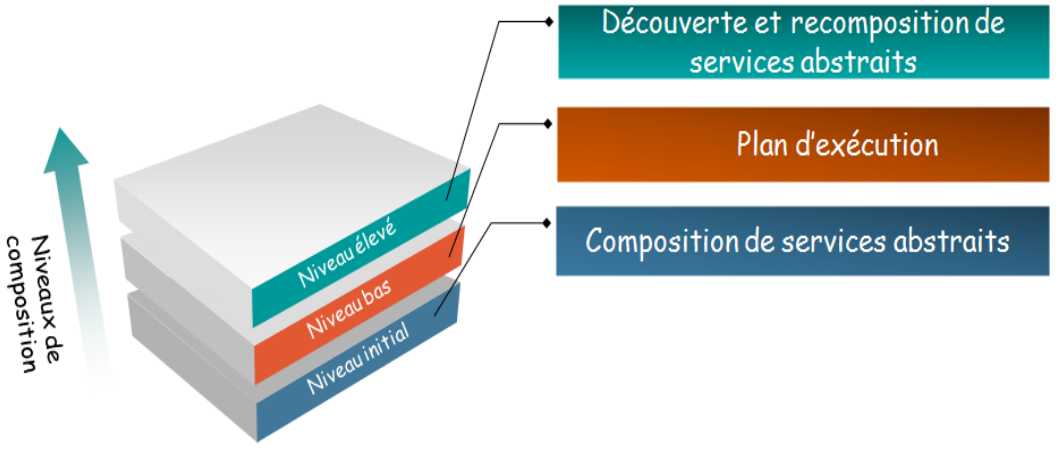


FIGURE 2.11 – Architecture en couche du processus de composition de services.

Niveau initial : Composition de services abstraits
A ce niveau un plan abstrait est construit en utilisant l'algorithme FCoSC (pour Feasibility and Construction of a Services Composition) [48].

Si la tâche demandée est satisfaite, le plan sera validé. Dans le cas contraire (erreur de service), un message est transmis au gestionnaire de tâches pour modifier cette tâche ou opter pour une autre.

Niveau bas : Exécution du plan
Ce niveau gère les défaillances d'exécution de service d'exécution en utilisant le mécanisme d'apprentissage. Pour chaque service abstrait du le plan de composition abstrait, l'objectif est de trouver

une action optimale en termes de QoS et de probabilité de réponse en utilisant l'algorithme QoSEA(pour Quality of Service Estimation Algorithm).

Une fois la réponse de l'action obtenue, sa probabilité est mise à jours en utilisant l'apprentissage Bayésien. Si cette réponse est positive, le système passe au service abstrait suivant, sinon il cherche une autre action. Une fois que le paramètre de précision d'apprentissage est dépassé sans trouver un service approprié, l'échec de composition est passé au niveau haut.

Niveau élevé : Découverte de services abstraits et recomposition

Une fois que toutes les actions d'un service abstrait ont échoué, ce niveau exécute l'algorithme SDT(pour Services Discovery for a Task) [48] pour découvrir un nouveau service abstrait dans le registre, le système essaie de construire un nouveau plan par le mécanisme de recomposition, l'apprentissage bayésien est aussi utilisé à ce niveau pour gérer la recomposition de services.

b) Bilan

Les auteurs ont proposé une approche de composition de services robotiques basée sur l'apprentissage Bayésien, afin de prendre en compte la nature stochastique de l'environnement, modélisé par une probabilité de réponse. Cette approche s'est également penchée sur l'aspect dynamique de l'environnement ubiquitaire, elle considère notamment le contexte d'utilisation et la qualité des services disponibles.

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.

2.3.2.6 Composition de service avec QdS dans les environnements dynamiques orientés services

a) Description

Les environnements de services sont connus pour être dynamique, ce qui 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 dégradation (dû par exemple à une déconnexion ou à une faible connectivité 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.

L'objectif de cet article est de faire face à la dynamicité des environnements de services lors de la sélection, la composition et l'exécution des services.

Cet article [57] présente un algorithme de sélection de services adéquat qui vise à réaliser l'objectif précité. L'algorithme consiste en une heuristique guidée. Ce choix est dû au fait que le coût de calcul introduit par l'approche basée sur les heuristiques est réduit, fournissant ainsi une solution en un temps satisfaisant qui vise à déterminer un ensemble de composition de services quasi-optimales, ces compositions doivent vérifier :

- Le respect des contraintes de QdS imposées par l'utilisateur sur la composition totale.
- Maximiser une fonction d'utilité QdS.

Lors de l'exécution de la composition de services, si un service de composition donné, n'est plus disponible ou qu'il a connu une dégradation en terme de QdS, une composition de remplacement sera exécutée. La construction des compositions de services se compose :

1. La phase de découverte ;
2. La phase de sélection ;
3. La phase de compositions de services.

En ce qui concerne la découverte de services, les auteurs adoptent une approche sémantique, telle que pour chaque activité (service composite) de la composition de service, la phase de découverte donne un ensemble de services candidats capables d'accomplir une tâche et de respecter les exigences de QdS des utilisateurs.

Pour la phase de sélection, les auteurs introduisent un algorithme heuristique basé sur les techniques de clustetring, qui permet le regroupement des services par rapport à leur QdS dans un ensemble de clusters qui feront références aux niveaux de QdS.

Une fois la sélection accomplie, la phase de composition utilise les services sélectionnés pour définir une composition de services exécutable.

1. Le modèle de QdS

Pour la spécification de la QdS dans les environnements dynamiques de services, les auteurs ont adopté un modèle sémantique de QdS formulé comme un ensemble d'ontologies qui permet de spécifier les attributs de QdS.

Les attributs de QdS sont divisés en 2 groupes :

– Les attributs quantitatifs (temps de réponse, disponibilité, fiabilité, débit).

– Les attributs qualitatifs (sécurité, confidentialité, etc.).

Dans cet article, les auteurs ne considèrent que les attributs quantitatifs de QdS, car les attributs qualitatifs peuvent être représentés comme des attributs quantitatifs à valeur booléenne. Les attributs quantitatifs sont à leurs tours divisés en 2 classes :

• **Attributs positifs** (*disponibilité, fiabilité, débit*) : ce sont les attributs qu'il faut maximiser ;

• **Les attributs négatifs** (*Temps de réponse, prix*) : ce sont les attributs qu'il faut minimiser.

Les valeurs de QdS sont déterminées de deux façons :

– Lors de la sélection de services ; ces valeurs sont données par les fournisseurs de services (par exemple, basés sur les exécutions précédentes des services ou à l'aide des commentaires des utilisateurs). Ils représentent la QdS annoncée ;

– Lors de l'exécution ; les valeurs de QdS sont fournies par un composant de surveillance pour permettre une évaluation ultérieure des services. Ils représentent la QdS d'exécution.

2. Modèle de composition

Les auteurs considèrent un ensemble de modèles couramment utilisé dans les approches de composition de services qui couvre la plus part des structures spécifiées par les langages de compositions (BPEL) :

◦ SEQUENCE : exécution séquentielle des services concrets ;
◦ ET : exécution parallèle des services concrets ;
◦ XOR : exécution conditionnelle des services concrets ;
◦ LOOP : exécution itérative des services concrets.

Calcul de la QdS des services composites

Pour chaque service abstrait, les auteurs représentent la QdS d'un seul service candidat Si en utilisant d'un vecteur QdS_{Si}=<$q_{i,1}$,...,$q_{i,n}$> où n représente le nombre d'attribut de QdS requis par l'utilisateur et qi,j représente la valeur de l'attribut de QdS j (1<= j<= n) du service i.

La composition de services est évaluée à base des vecteurs de QdS des services composites tenant compte des modèles de composition. L'estimation de la QdS, elle est basée sur approche pessimiste qui considère les pires valeurs de QdS pour chaque modèle de composition.

Attributs de QdS	Modèle de composition			
	SEQUENCE	ET	XOR	LOOP
Temps de réponse	$\sum_i^n$=1rt$_i$	Max(rt$_j$)	Max(rt$_j$)	Rt*K
Fiabilité	$\prod_i^n$ =1re$_i$	$\prod_i^n$=1re$_i$	Min(re$_j$)	re$_k$
Disponibilité	$\prod_i^n = 1re_i$	$\prod_i^n = 1re_i$	$Min(av_j)$	av_k
Débit	$Min(th_j)$	$Min(th_j)$	$Min(th_j)$	th
Prix	$\sum_i^n = 1Pi$	$\sum_i^n = 1Pi$	Max (P$_i$)	P*k

TABLE 2.2 – Matrice de classification de services concrets

3. Algorithme de sélection de services

Les algorithmes de sélection révèlent deux approches générales :

- Sélection locale.
- Sélection globale.

La sélection globale, qui contrairement à la sélection locale, assure la satisfaction de toutes les contraintes de QdS, cette approche considère toutes les compositions possibles de services et sélectionne la meilleure.
Les auteurs proposent ainsi un algorithme heuristique qui combine les techniques de sélection locales et globales. L'algorithme procède comme suit :

- **Phase de mise à l'échelle** : est une phase de prétraitement visant à normaliser les valeurs de QdS associées à des attributs positifs et négatifs de QdS en les transformant en une valeur comprise entre 0 et 1.

- **Classification locale** : est une phase qui vise à classer les services candidats selon les différents niveaux de QdS. Elle est réalisée localement pour chaque activité de la composition de services abstraits ; chaque niveau contient l'ensemble des services candidats ayant à peu prés la même QdS, pour ce faire les auteurs ont utilisés des techniques de clustering, notamment l'algorithme des K-means.

- **La sélection globale** : vise à sélectionner les compositions quasi-optimales qui respect les contraintes de QoS maximise la fonction d'utilité.

b Bilan

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

lors de l'exécution de la composition. Cette approche fait face à cette dynamicité des environnements de services lors de la sélection, la composition et l'exécution des services.

2.3.2.7 Composition dynamique de services sensible au contexte en environnement ubiquitaire

a Description

Ce travail [52] propose un modèle de composition de services qui repose sur les mêmes couches de [49] :
Première couche : elle consiste à déterminer le plan abstrait de la composition, en fonction de la requête de l'utilisateur.
Deuxième couche : cette approche adopte un mécanisme d'apprentissage basé sur l'estimation de la réputation du service abstrait, ainsi que les paramètres de qualité.

Estimation de la réputation

$$Rt(a) = Nb_{succes}/Nb_{selection} \tag{2.3}$$

où
Rt(a) : est la réputation du service concret (a) à l'instant.

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 de services indiqués dans la figure ci-dessous.

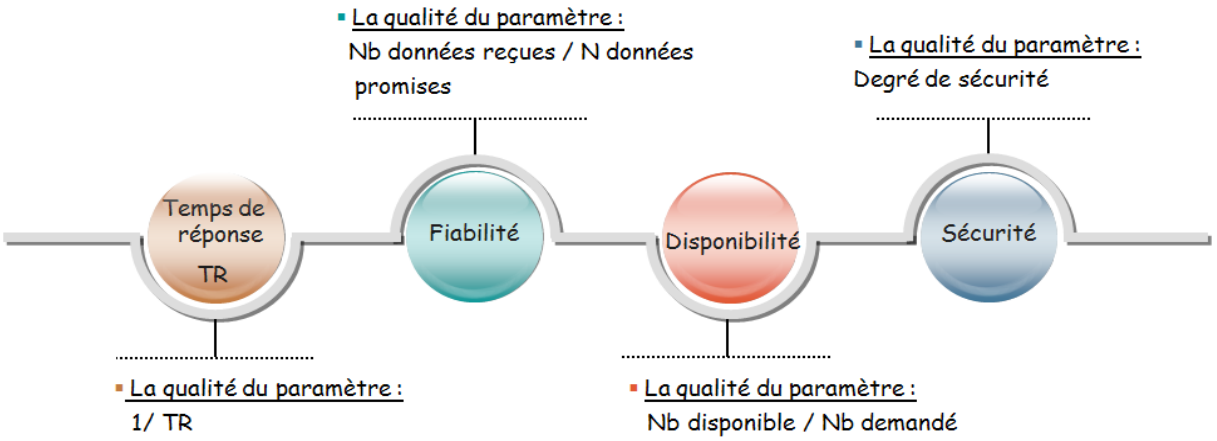


FIGURE 2.12 – Paramètre de qualité de services.

Remarque : selon un mécanisme d'apprentissage, la QdS du service concret et la réputation du service abstrait sont estimées après chaque fin d'exécution.

Pour chaque service concret, on estime la qualité de son expérience comme suit :

$$QoT_t(s) = \sum_{i=1}^{NbQ}(W_k^i \times QoS_t^i(s)), W_k^i = \frac{(WUP_k^i + WUC_k^i + WEC_k^i)}{N} \tag{2.4}$$

Où :
$QoT_t(s)$: qualité I du service concret S à l'instant t.
NbQ : le nombre de paramètres de qualité des services.
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 : poids de la qualité i pour l'utilisateur k en fonction de son contexte.
WEC_k^i : poids de la qualité i pour l'utilisateur k en fonction du contexte de son environnement.
N : le nombre d'informations collectées (WUP_k^i, WUC_k^i, WEC_k^i).

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

$$CS_k = ArgMax(QoE_t(x))_{x \in AS_j^{CS}} \wedge CS_k > QT \tag{2.5}$$

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

b) Bilan

Cette approche s'est focalisée sur l'aspect dynamique de l'environnement ubiquitaire, elle considère notamment 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 dernières exécutions mais prédits à partir du contexte courant.

La dynamicité des environnements de services pose un problème lorsqu'un ou plusieurs services qui font parties d'une composition de services ne sont plus disponibles ou bien que leur QdS diminue lors de l'exécution de la composition. En effet, 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.

2.3.3 Synthèse et comparaison entre les approches étudiées

La composition de services dans les environnements ubiquitaires est un domaine de recherche récent qui s'inspire des recherches menées conjointement sur les services web. Plusieurs approches ont été proposées pour résoudre ce problème, et la plupart d'entres elles visent à intégrer les aspects suivants : automatisation du processus de composition des services, prise en compte de la nature stochastique et dynamique des environnements ubiquitaires, et intégration de la sensibilité au contexte d'utilisation. C'est dans cette optique que s'inscrivent les travaux de recherche menés par les auteurs des approches étudiées dans ce chapitre. Chacune de ces approches présentent des points forts dans certains aspects de la composition mais aussi quelques lacunes.

Les approches passées en revue sont centralisées autour d'un moteur de composition, cela découle du principe même d'orchestration de services qu'elles implémentent. Ce qui rend le système susceptible de tomber sur un éventuel point de défaillance centrale.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 la dynamicité 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 requête 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.

Dans un environnement ubiquitaire, le contexte est riche et varié ; aussi, il mérite d'être pleinement pris en considération par les dispositifs évaluant la performance des systèmes. Dans les approches [37, 47, 51, 52] étudiées, cela a été pris en compte dans le processus de sélection et de composition de services. Contrairement aux approches [48, 57, 49] où cette relation entre le contexte et la sélection n'a pas été pleinement considérée. De plus [51, 52], ont un modèle de sélection basé sur des paramètres de QdS estimés à partir des dernières exécutions.

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 inadéquat de se reposer sur les dernières exécutions pour estimer le comportement que le service aura.

Les approches étudiées pour la composition se diffèrent par la technique utilisée pour la composition, leur degré d'automatisation (manuelles, interactives, automatiques), leur prise en compte de la sémantique (basées sur les ontologies ou non), du contexte, et de la QoS, également par leur caractère dynamique ou statique. Le tableau suivant résume les caractéristiques citées ci-dessus des différentes approches de composition de services qu'on a passés en revue dans un environnement ubiquitaire :

Approche	Technique	Contexte	QoS	Distribué	Automatique	Sémantique	Dynamique	Tolérance aux pannes
Vers une découverte de services orientée contexte et un modèle de composition (CBsec) [37]	SMA	✓			✓		✓	
Composition flexible de services d'objets communicants pour la communication ambiante [47]	SMA			✓	✓	✓	✓	
Vers une approche automatique de composition de services en robotique ubiquitaire [48]	Planifiacation				✓		✓	✓
Les règles de composition automatique de services en environnement ubiquitaire [49]	Planifiacation		✓		✓	✓	✓	✓
Modèle de composition de services basé sur la QdS en robotique ubiquitaire [51]	Planifiacation	✓	✓		✓		✓	✓
Composition dynamique de services sensible au contexte en environnement ubiquitaire [52]	Planifiacation	✓	✓		✓	✓	✓	✓
Composition de service avec QdS dans les environnements dynamiques orientés services [57]	Planification		✓		✓	✓	✓	✓

TABLE 2.3 – Tableau compratif

2.3.4 Conclusion

L'informatique ubiquitaire, est un paradigme récent de l'informatique personnelle dont l'objectif est de permettre aux utilisateurs de consulter des données n'importe quand, n'importe où à travers leurs dispositifs d'accès.

Cependant, le nombre et la diversité des objets communicants posent la question du temps et de l'attention que les utilisateurs devront leur accorder pour interagir et ceci en sollicitant le moins possible l'attention des usagers. Un utilisateur nomade est souvent situé dans un environnement très dynamique. Les utilisateurs peuvent changer d'endroit ce qui éventuellement peut modifier l'exécution de leurs tâches en fonction des caractéristiques contextuelles de l'endroit où ils se trouvent.

La prise en compte d'informations contextuelles ainsi que des préférences des utilisateurs est d'une importance capitale dans la réalisation de cette tâche. La méthode de composition doit alors permettre la sélection et la composition automatique des services qui répondent à la requête de l'utilisateur tout en respectant des contraintes précises. Et là réside toute la complexité de la composition de services dans l'environnement ubiquitaire.

Dans ce chapitre, nous avons dressé une étude de quelques approches proposées pour résoudre le problème de composition de services dans l'environnement ubiquitaire. Nous avons aussi effectué une classification des méthodes de composition de service, selon l'interaction entre les services et l'intervention de l'utilisateur.

CHAPITRE 3

APPROCHE DE COMPOSITION DYNAMIQUE DE SERVICE AVEC QDS DANS LES ENVIRONNEMENTS UBIQUITAIRES.

3.1 Introduction

Dans ce chapitre, nous décrivons notre contribution portant sur la composition dynamique de services avec QdS dans les environnements ubiquitaires. En premier lieu, nous mettons l'accent sur la nécessité de modéliser le contexte, et pour cela, le langage de modélisation UML a été utilisé pour la représentation des connaissances se rapportant aux profils et préférences de l'utilisateur ainsi son environnement. L'acquisition et le traitement des informations contextuelles jouent un rôle fondamental dans le développement des systèmes adaptatifs. En effet, s'adapter au contexte de l'utilisateur est un défi essentiel, gage du confort de l'utilisateur, mais aussi de leur attractivité et donc de leur pérennité.

On a vu primordiale de considérer et de définir ce que l'on entend par qualité de service (Quality of Service : QoS), plus particulièrement les attributs manipulés, qui représentent l'ensemble des phénomènes pouvant influencer les performances du service qui déterminent le degré de satisfaction de l'utilisateur.

En second lieu, nous décrivons l'architecture générale de la composition de services orientée QdS et sensibles au contexte. Cette architecture comporte plusieurs modules qui collaborent entre eux afin de fournir un service adapté aux besoins de l'utilisateur.

En dernier, nous détaillons l'approche proposée en mettant certaines hypothèses qui nous serviront d'appui pour le bon fonctionnement du système proposé.

3.2 Modélisation du contexte

L'un des problèmes majeurs des systèmes pervasifs concerne l'adaptation au contexte d'utilisateur.

Unified Modeling Language (UML) est un moyen de représenter la connaissance. Cette représentation correspond à "une spécification explicite et formelle d'une conceptualisation partagée " [64].

Notre choix s'est porté sur l'UML pour la représentation du contexte. Ce choix se justifie par le fait que l'UML est considérée comme étant un modèle de représentation expressif. Il permet de représenter la sémantique existante entre différents paramètres qui forment le contexte ainsi que leurs relations. Ainsi, nous avons défini les classes de contexte suivantes : le contexte de l'utilisateur, le contexte de l'environnement, et le contexte du service, et le contexte de ressource. La description des quatre classes du contexte est définie par la figure (3.1) :

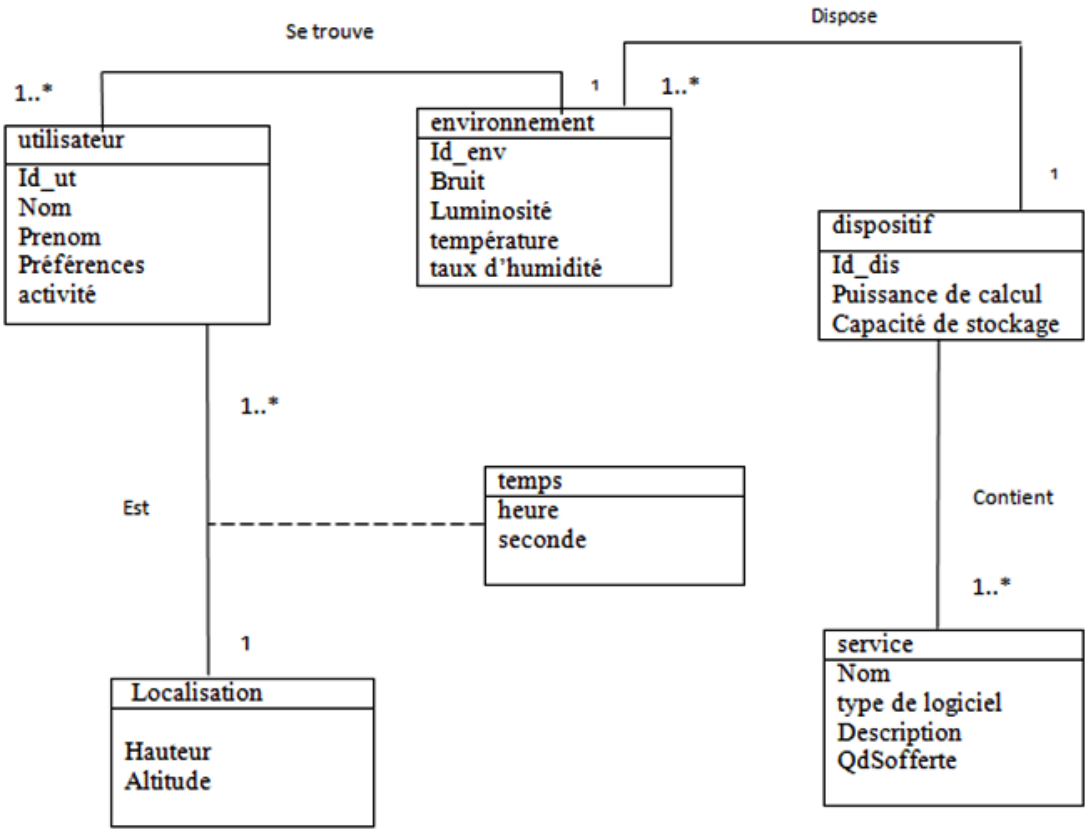


FIGURE 3.1 – Modélisation du contexte en utilisant UML.

3.2.1 Le contexte de l'utilisateur

Il s'agit de toutes les informations relatives à l'utilisateur telles que : ces préférences (centres d'intérêts), les informations indiquant son état physique ou émotionnel (la température corporelle, le rythme cardiaque, etc.), sa localisation, son activité courante, etc.

3.2.2 Le contexte du service

Cette classe représente toute information non fonctionnelle du service telles que : son coût, sa durée d'exécution, sa disponibilité, sa fiabilité, et sa réputation (renommé du fournisseur du service).

Ce contexte représente, en quelque sorte, les informations considérées par le modèle de qualité de service.

3.2.3 Le contexte de l'environnement

Il décrit les informations sur l'environnement de l'utilisateur telles que : le climat, la température, la luminosité, le bruit, et quelques données sur la situation entourant l'utilisateur (les personnes ou objets à proximité), etc.

3.2.4 Le contexte de ressource

Il contient toutes les informations sur les équipements (ressources) dont dispose l'utilisateur tels que : le terminal utilisé, le réseau de communication, les appareils qui assistent l'utilisateur dans sa vie quotidienne. Ainsi les propriétés de ces équipements telles que : les dispositifs d'affichage, la puissance de calcul, la capacité de stockage, l'énergie, le débit du réseau, etc.

3.3 Qualité de services

Avec la croissance de la mondialisation, le besoin d'accès continu à l'information de façon robuste et flexible à travers des services se fait sentir. Ce mode d'accès à l'information devient un nouveau paradigme de programmation et d'organisation des opérations. Il s'agit ici de répondre aux besoins évolutifs de l'utilisateur tout en prenant en considération la nature dynamique du contexte et la qualité des services disponibles.

Il s'agit ainsi de traiter la dynamicité et l'incertitude de l'environnement ubiquitaire avec pour objectif d'assurer la continuité et la qualité des services conformément aux besoins exprimés par l'utilisateur. L'objectif ici est de permettre la satisfaction efficace des besoins fonctionnels et non-fonctionnels (QdS), en respectant les limites posées par les contraintes de temps et de disponibilité des ressources tout en optimisant le nombre de services mis en jeu et le temps de composition. L'enjeu des systèmes pervasifs est, dans cette perspective, de fournir les cadres méthodologiques à même de permettre une utilisation fiable, pertinente et efficace de ces systèmes.

La QdS est donc l'aptitude d'un service à répondre adéquatement à des exigences, exprimées ou implicites, visant à satisfaire les utilisateurs. Ces exigences peuvent être liées à plusieurs aspects d'un service à savoir : la disponibilité, la fiabilité, le temps de réponse, la sécurité, etc.

Par ailleurs, établir et s'assurer de la QdS représente un enjeu crucial puisque ceci permet d'établir une relation de confiance entre le fournisseur d'un service et un client en attente d'une certaine qualité.

Dans notre travail, nous nous intéresserons aux paramètres de la QdS décrits dans le tableau(3.1) :

Paramètres	Descriptions
Temps de réponse	Le temps de réponse du service est l'intervalle entre le moment où un utilisateur envoie une requête et l'instant où il reçoit une réponse.
Fiabilité	La fiabilité est l'aptitude d'un service à accomplir une fonction requise dans des conditions données pour une période de temps donnée. C'est la probabilité de n'avoir aucune défaillance à l'instant t.
Disponibilité	Le service devrait être prêt pour une consommation (exécution) immédiate lors d'une invocation.
Sécurité	Il existe plusieurs traitements permettant de sécuriser les communications entre services (Authentification, Autorisation, Confidentialité, Non répudiation).
Temps critiques	L'exécution du service doit s'effectuer en un temps inférieur à une borne maximale. Un système temps réel est un système dont l'exactitude des résultats ne dépend pas seulement de l'exactitude logique des calculs mais aussi de la date à laquelle le résultat est délivré [54].

TABLE 3.1 – Les attributs de qualité de services utilisés dans l'approche proposée.

Les attributs de QdS sont divisés en deux classes [57] :

Les attributs positifs qui sont les attributs qu'il faut maximiser (disponibilité, fiabilité, sécurité, temps critique) et *les attributs négatifs* qui représentent les attributs qu'il faut minimiser (Temps de réponse).

Les valeurs de QdS pour un service, sont déterminées de deux façons[57] :

- *Lors de la sélection du service* ; ces valeurs sont données par les fournisseurs de services (par exemple, basés sur les exécutions précédentes des services ou à l'aide des commentaires des utilisateurs). Ils représentent la QdS annoncée ;
- *Lors de l'exécution* ; les valeurs de QdS sont fournies par un composant de surveillance pour permettre une évaluation ultérieure des services. Ils représentent la QdS d'exécution.

3.4 Représentation de services

3.4.1 Services abstraits

Un service abstrait regroupe plusieurs services concrets. Ces derniers offrent la même fonctionnalité et possèdent les mêmes entrées et sorties. En d'autre terme, un service abstrait représente une classe de services concrets d'un domaine particulier. Un service abstrait i est noté SA_i, il est décrit par un quadruplet <SA_i^{in} ,SA_i^{out} , SA_i^{cs}, R> [49] tels que :

- SA_i^{in} : Ensemble des entrées de SA_i ;
- SA_i^{out} : Ensemble des sorties de SA_i ;
- R : la réputation du service abstrait ;
- SA_i^{cs} : Ensemble des services concrets de SA_i
- tel que :
- SA_i^{cs} := {cs$_{i,1}$, $cs_{i,2}$,..., $cs_{i,n}$} et $\forall$ $cs_{i,j}$, $cs_{i,k} \in SA_i^{sc}$ ($cs_{i,j}^{in}$= $cs_{i,k}^{in}$ = SA_i^{in})$\wedge$(cs$_{i,j}^{out}$ = $cs_{i,k}^{out}$ = SA_i^{out}).

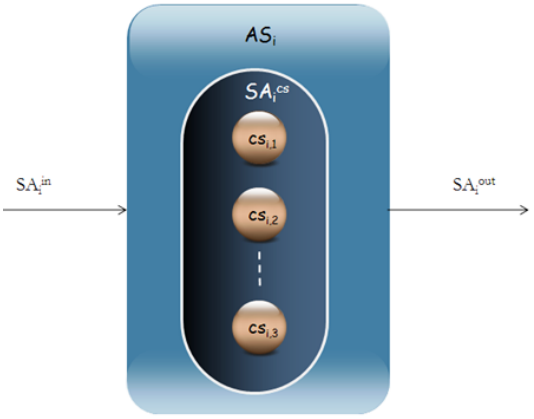


FIGURE 3.2 – Représentation d'un service abstrait.

3.4.2 Services concrets

Un service concret est un service qui réalise une certaine fonctionnalité en agissant sur des données en entrées afin de produire des données en sortie. Dans le cadre de notre travail nous avons introduit un sixième composant aux cinq déjà proposé dans [48] à savoir le facteur d'exploitation. Un service concret, noté CSi, est décrit par
(CSiin,CSiout, Prec, Eff, QdS, Fi) Tels que :

CS$_i^{in}$: sont les paramètres d'entrées du service ;
CS$_i^{out}$: sont les paramètres de sortie du service ;
Prec : les prés-conditions du service ;
Eff : les effets du service ;
QdS : les paramètres de qualité du service ;
F$_i$: le facteur d'exploitation du service CS$_i$, ce facteur représente le nombre d'invocations où le service a répondu. Cette réponse peut survenir après le délai quand il s'agit d'une requête temps réel ; le service ne peut également répondre qu'à certains paramètres de sorties requis.

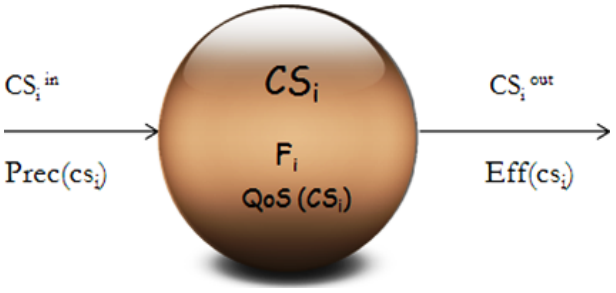


FIGURE 3.3 – Représentation d'un service concret.

3.5 Architecture de composition de services

La figure (3.4) représente l'architecture générale sur laquelle repose notre système de composition de services sensible au contexte.

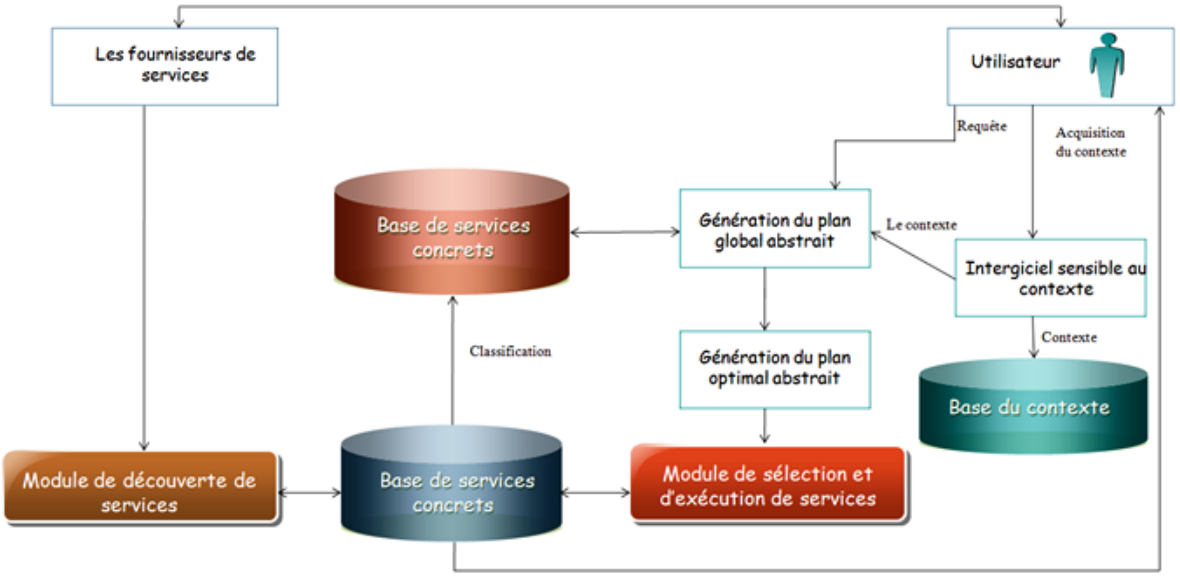


FIGURE 3.4 – Architecture du système de composition de services

L'utilisateur exprime ses besoins à travers une requête, cette dernière sera traitée par les modules suivants :

3.5.1 Le module de découverte de services

Ce module explore les services proposés par les fournisseurs de services et les stocke dans la base de services concrets. Il effectue également une recherche périodique afin de tenir compte des modifications au niveau des services et mettre à jour la base.

3.5.2 L'intergiciel sensible au contexte

Il récupère et traite le contexte et le stocke dans la base du contexte.

3.5.3 La base de services concrets

Elle contient les services concrets mis à disposition et découvert par le module de découverte de services.

3.5.4 La base de services abstraits

Elle contient les services abstraits, qui seront utilisés pour la génération du plan abstrait. Chaque service abstrait représente la classe des services concrets possédant la même fonctionnalité.

3.5.5 Le module de sélection et d'exécution de services :

Il s'occupe de la sélection des services concrets les plus appropriés vérifiant les conditions exigées dans les deux phases proposées dans notre approche (présélection et sélection finale) et constituant le plan d'exécution qui sera exécuter par la suite selon un ordre établis préalablement.

3.5.6 La base du contexte :

Elle contient les informations contextuelles, fournies par un intergiciel sensible au contexte, relatives à l'utilisateur et son environnement.

3.6 Approche pour la composition dynamique de services avec QdS

3.6.1 Hypothèses :

Nous proposons une approche de composition dynamique de services qui se base sur les hypothèses suivantes : Les services concrets disponibles sont découverts par le module de découverte de services et mis à disposition pour une éventuelle utilisation, dans le répertoire des services concrets CSD (*Concret Service Directory*).Les services concrets sont classés en fonction de leurs fonctionnalités dans les services abstraits. Les deux répertoires sont régulièrement mis à jours en fonction de la disparition ou de l'apparition des services concrets.

3.6.2 Principe de fonctionnement de l'approche proposée :

3.6.2.1 La génération du plan abstrait global :

Elle consiste en la génération automatique d'un plan global abstrait. Le plan contient tous les services abstraits possibles qui pourraient contribuer à composer le service demandé exprimé dans

la requête de l'utilisateur.

Pour construire 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. Plus précisément, les étapes suivantes sont exécutées [5] :

- Sélectionner tous les services abstraits ASn qui fournissent tous ou une partie des paramètres de sortie requis.
- Pour chaque service sélectionné ASn, une nouvelle sélection est faite pour trouver l'ensemble des services abstraits qui génèrent les paramètres d'entrée de AS_n. 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.

3.6.2.2 Génération du plan optimal abstrait :

Le plan global généré sera utilisé, d'une part pour générer un plan abstrait optimale et d'autre part pour régénérer un nouveau plan optimal en cas d'erreurs [49].

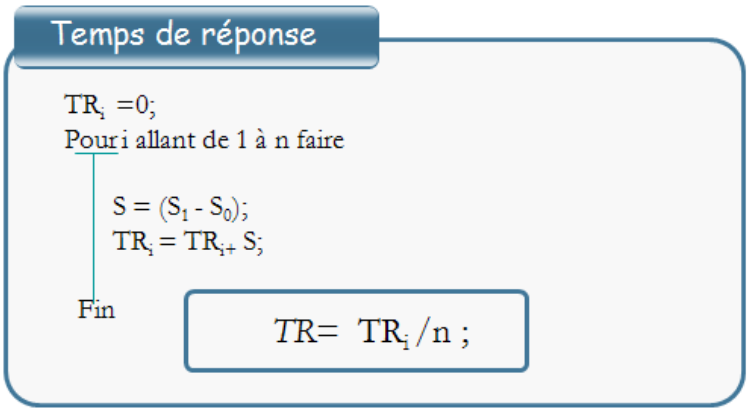
L'optimisation du plan globale est basée sur la réputation et la complémentarité des paramètres fournis par les services abstraits, afin d'éviter la sélection des services qui offrent les mêmes paramètres [49]. La réputation du service abstrait est donnée par :

$$Rt(a) = Nb_{succes}/Nb_{selection} \tag{3.1}$$

où
Rt (a) : est la réputation du service concret (a) à l'instant t.
Par conséquent, le plan optimal généré à un nombre de services plus réduit pouvant participer à la composition du service exprimé dans la requête T.
Après la génération du plan optimal, un service concret sera sélectionné pour chaque service abstrait selon les paramètres de qualité de service et le contexte.
Dans cette section nous détaillons la manière dont les paramètres de qualité de services sont calculés :

1. **Le temps de réponse** :
   Le temps de réponse (noté TR) d'un service concret mesure l'intervalle de temps entre le moment de l'envoi de la requête et le moment où la réponse est reçue. Dans notre travail, le

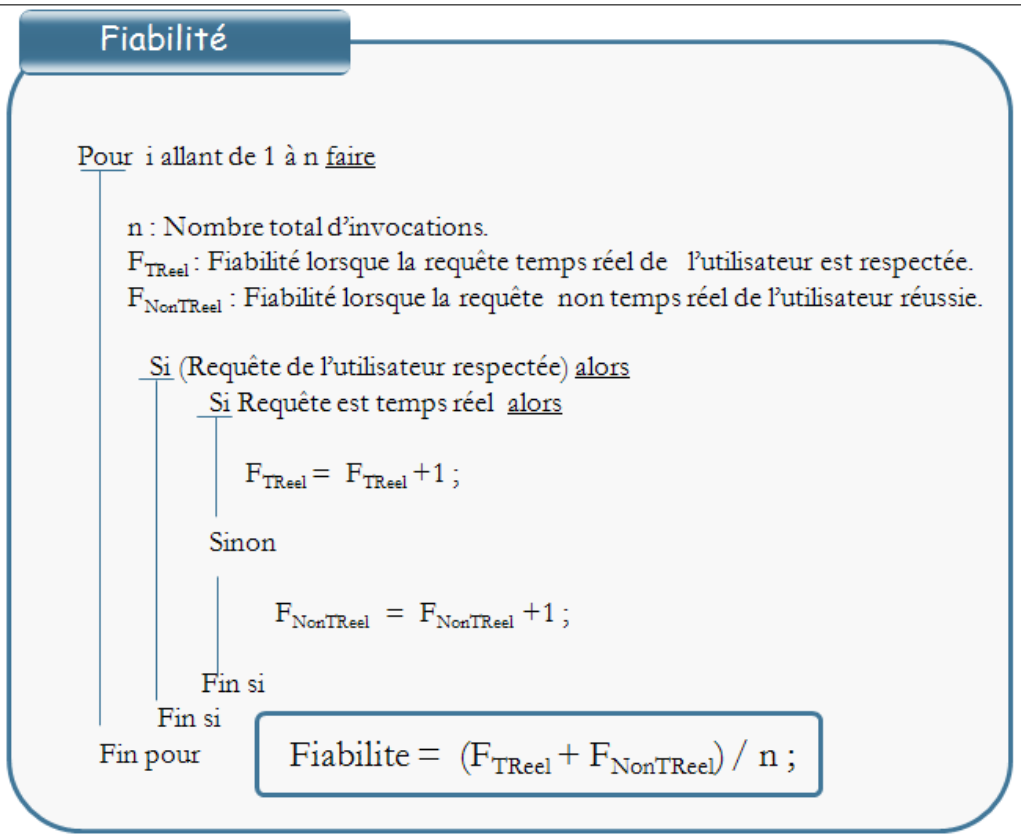
temps de réponse est calculé en se basant sur les exécutions passées. L'algorithme ci-dessous illustre la manière dont le temps de réponse est estimé.



Où

- n : Nombre total d'invocations pour un service concret i.
- s_0 : l'instant de l'envoi de la requête par l'utilisateur.
- S_1 : l'instant de réponse du service concret i.
- TR_i : est l'ensemble des observations passées du temps de réponse du service concret CS.
- TR : est l'ensemble des observations passées du temps de réponse sur le nombre total d'invocations.

2. **La fiabilité** :
   dans le cadre de notre travail, ce paramètre (noté fiabilité) représente le nombre d'invocations où le service a répondu avec toutes les sorties requises et dans les délais (lorsque la requête est temps réel) sur le nombre total d'invocations.

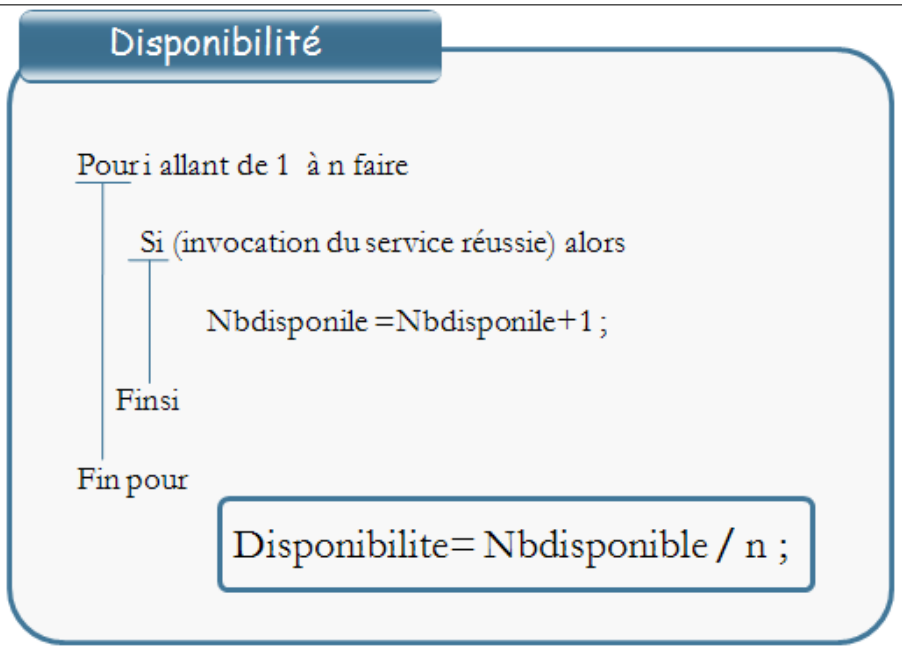


Où

- n : Nombre total d'invocations pour un srvice concret i.
- FT_{Reel} : Fiabilité lorsque la requête temps réel de l'utilisateur est satisfaite.
- $F_{NonTReel}$: Fiabilité lorsque la requête non temps réel de l'utilisateur a réussi.

3. **La disponibilité** :

   La disponibilité (notée disponibilite) représente le nombre de fois où le service est accessible sur le nombre total d'invocations, sa valeur est estimée de la manière suivante :

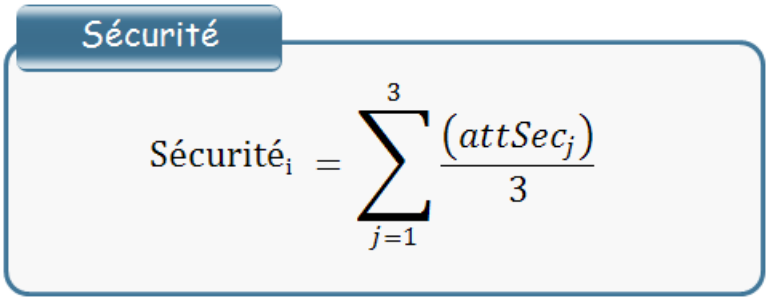


Où

– n : Nombre total d'invocations.
– Nbdiponible : le nombre de fois que le service est disponible.

4. **La sécurité** :

Il représente le degré de sécurité du service. La question de sécurité est très importante, dans la mesure où elle peut être exigée par les utilisateurs. Cette sécurité vise à réaliser et à garantir les objectifs suivants : la confidentialité, l'authentification, et l'intégrité. La sécurité est mesurée en fonction des objectifs qu'elle vise à atteindre, et cela est estimé comme suit :



Tel que, $attSec_j$ est l'attribut de sécurité j du service concret i. Ces attributs sont au nombre de trois : confidentialité, authentification, et intégrité.

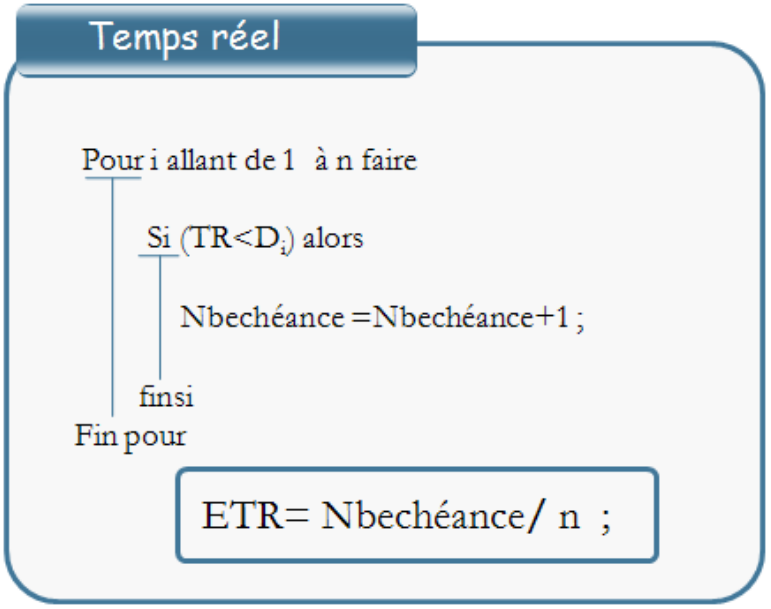
– Si $Scurit_i$ =j alors on dira que le service concret i est conforme à l'objectif visé par la sécurité ;
– Si $Scurit_i$ =3 alors on pourra dire que le service concret i est conforme aux trois objectifs ciblés $\forall j \in \{1, 2, 3\}$.

5. **La Contrainte temps réel** :

La nature dynamique des environnements dans lesquels les systèmes s'exécutent soulève des défis majeurs, notamment le comportement de ces systèmes qui pourraient être critiques et

complexes.

Le temps critique permet de prendre en compte le moment où le résultat est produit, il représente le temps de réponse maximum que le service ne doit pas dépasser. La valeur de l'échéance temps réel est calculée comme suit :



Où
- n : le nombre total d'invocations.
- TR : le temps de réponse du service.
- D_i : échéance temporelle (de chaque service concret CS_i).
- Nbechéance : le nombre d'invocations où le service a respecté la contrainte temps réel.

Dans le cadre d'applications temps réel, le non-respect d'une contrainte temporelle peut avoir des conséquences plus ou moins chaotiques. C'est par exemple le cas dans une application de robot mobile autonome capable de détecter des obstacles ; la détection tardive d'un obstacle, se trouvant sur sa trajectoire, peut ici aboutir à une collision susceptible d'endommager le robot.

Comme nous l'avons indiqué précédemment, les valeurs de QdS peuvent être négatives ou positives. En effet, certaines valeurs des attributs de QdS doivent être minimisées, alors que d'autres doivent être maximisées. Pour faire face à ce problème, une phase de mise à l'échelle doit être effectuée pour normaliser les valeurs des attributs de QdS.

Phase de mise à l'échelle

La mise à l'échelle est une phase de prétraitement visant à normaliser les valeurs de QdS associées à des attributs positifs et négatifs en les transformant en une valeur comprise entre 0 et 1. Pour chaque service abstrait, on représente la QdS d'un service candidat CS_i en utilisant un vecteur QdS_{CS_i}=<$q_{i,1}$,...,$q_{i,n}$>

où

n : représente le nombre d'attributs de QdS requis par l'utilisateur

$q_{i,j}$: représente la valeur de l'attribut de QdS j (1<= j<= n) du service i.

Pour normaliser les valeurs des attributs de QdS, on utilise les formules suivantes :

Attributs négatifs :

$$q'_{i,j} = \begin{cases} \frac{q_j^{max} - q_{i,j}}{q_j^{max} - q_j^{min}}, & \text{if } q_j^{max} - q_j^{min} \neq 0\,; \\ 1, & \text{else.} \end{cases} \tag{3.2}$$

Attributs positifs :

$$q'_{i,j} = \begin{cases} \frac{q_{i,j} - q_j^{min}}{q_j^{max} - q_j^{min}}, & \text{if } q_j^{max} - q_j^{min} \neq 0\,; \\ 1, & \text{else} \end{cases} \tag{3.3}$$

Où

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

3.6.2.3 Sélection du meilleur service concret

Dans le but de répondre aux exigences de l'utilisateur lors de la composition de plusieurs services atomiques, il est nécessaire que ces exigences soient spécifiées de façon claire et explicite. Les exigences de l'utilisateur, en terme de qualité, peuvent être locales (à satisfaire par chaque service candidat) ou globales (à satisfaire par le service composite).

La sélection locale : est la qualité exigée par l'utilisateur au niveau de chaque service concret, elle est exprimée comme un seuil QSe. Tout service n'offrant pas cette qualité sera rejeté de la liste des services qui seront probablement sélectionnées dans la composition.

La sélection globale : les exigences globales sont à satisfaire par le service composite. L'utilisateur spécifie donc les exigences globales sur un ensemble d'attributs de QdS. Dans notre approche de composition, on se limite à un seul attribut de QdS qui est le temps de réponse.

La qualité globale est définie comme étant un seuil QSeG. Par exemple : temps réponse global <230 ms, c'est-à-dire le temps de réponse global (toute la composition) doit être inférieur à 230ms.

La sélection du meilleur service concret se base sur la QdS et les attributs du contexte. Cette sélection se déroule en trois étapes :

La première, est une phase de présélection qui consiste à élire les services concrets dont la qualité de l'expérience (QdE) dépasse le seuil (QSe) spécifié par l'utilisateur. Ce seuil (qualité locale) représente l'exigence minimale sans laquelle l'utilisateur n'est satisfait. La qualité de l'expérience

QdE représente la qualité d'un service concret en fonction des préférences de l'utilisateur. En d'autre terme, la QdE est une somme pondérée des paramètres de qualité de service (QdS). Le poids associé à chaque paramètre est mesuré en fonction des préférences de l'utilisateur (indiqué dans son profil), son contexte, et le contexte de l'environnement. Ce poids reflète le degré d'importance d'un attribut donné par rapport à l'utilisateur.

La deuxième phase est la phase de sélection finale, elle adopte le facteur d'exploitation pour évaluer le degré de réponse de chaque service.

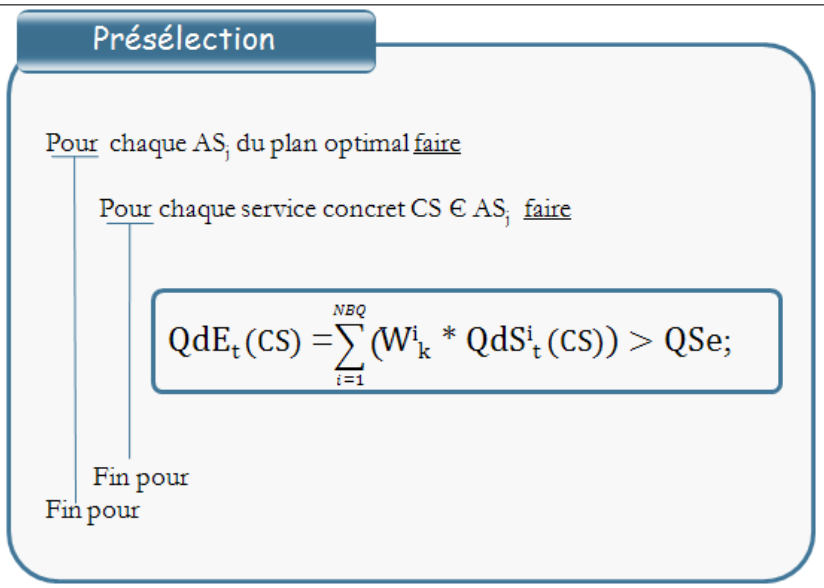
Notre approche apporte un plus quant à la contrainte temps réel, elle vise à assurer la satisfaction des besoins de l'utilisateur, et ce dans les délais. Ce délai représente le temps maximal que le service ne doit pas dépasser pour délivrer le résultat attendu. Si ce délai est écoulé, la réponse devient inutile dans certains cas (ce que l'on appelle contrainte molle), et chaotique dans d'autres (ce que l'on appelle la contrainte dure).

Dans la troisième étape une vérification sera faite sur le plan d'exécution. En effet, lorsque les services concrets constituant ce plan sont combinés ensemble, ils doivent répondre à la contrainte de la qualité global QSeG. Ce paramètre représente l'exigence en terme de temps de réponse exprimé par l'utilisateur. En d'autre terme, il représente la durée maximum que le temps d'exécution de l'ensemble des services combinés (service composite) ne doit impérativement pas dépasser.

Dans notre travail, nous utiliserons la sélection locale pour les deux phases : présélection et sélection finale, et la sélection globale pour la vérification de la faisabilité du plan de composition.

→ **Phase de présélection**

L'algorithme suivant permet de sélectionner les services concrets dont la qualité QdS dépasse le seuil (qualité locale) accepté (noté QSe) par l'utilisateur.



Où

- NBQ : représente le nombre de paramètres de QdS ;
- $QdE_t(CS)$: représente la qualité d'expérience du service concret CS à l'instant t ;
- QdS^i_t (CS) : représente la qualité "i" du service concret CS à l'instant "t".
- QSe : représente l'exigence minimale sans laquelle l'utilisateur n'est satisfait (qualité exigée localement, c-à-d au niveau de chaque service concret).
- W^i_k : représente le poids attribué par l'utilisateur "k" à l'attribut de qualité "i", il est calculer comme suit :

$$W^i_k = (WUP^i_k + WUC^i_k + WEC^i_k) / N$$

Où :

- WUP^i_k : représente le poids de la qualité "i" de l'utilisateur k en fonction de ses préférences.
- WUC^i_k : représente le poids de la qualité "i" de l'utilisateur k en fonction de son contexte.
- WEC^i_k : représente le poids de la qualité "i" de l'utilisateur k en fonction de son environnement.

Les sorties de cette phase constituent les entrées de la phase suivante.

2) **La sélection finale**

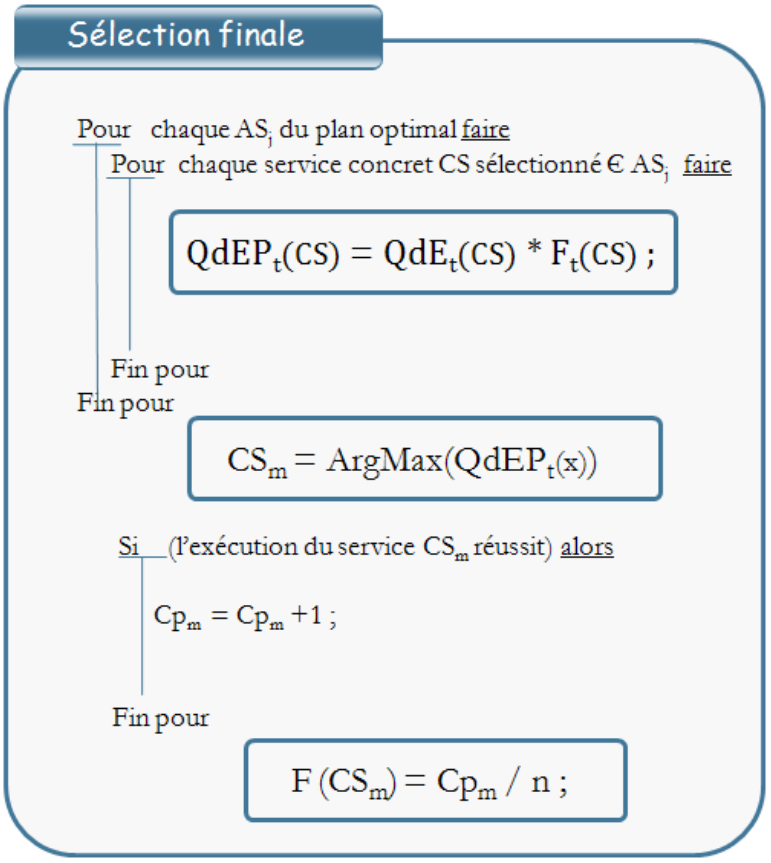
Dans cette phase, nous avons introduit le facteur d'exploitation permettant d'estimer le degré de réponse d'un service concret. En effet, ce facteur est évalué et mis à jours de la manière suivante :

- Il diminue dans le cas où le service n'a fournit aucun paramètre requis en sorties.
- Il augmente dans le cas où le service a répondu soit :
  - Lorsqu'il s'agit d'une requête non temps réel :

  - Le service a répondu avec quelques paramètres de sorties requis.
  - Le service a répondu avec tous les paramètres de sorties requis.
- Lorsqu'il s'agit d'une requête temps réel :
  - Le service a répondu avec quelques paramètres de sorties requis après ou dans les délais.
  - Le service a répondu avec tous les paramètres de sorties requis après ou dans les délais.

Dans notre approche, nous avons introduit un autre type de qualité, il s'agit de la qualité d'exploitation (notée QdEP). Elle est estimée en fonction du facteur d'exploitation (noté F) et de la qualité d'expérience (notée QdE) déjà calculée dans la phase de présélection. La fonction ArgMax() intervient pour trouver le meilleur service concret qui maximise la qualité d'exploitation.
Ce service participera dans la composition du service. Après exécution du service, le facteur d'exploitation sera mis à jour.

L'algorithme suivant décrit la phase de sélection finale.



Où

- $QdEP_t(CS)$: représente la qualité d'exploitation du service CS à l'instant t.
- $F_t(CS)$: représente le facteur d'exploitation du service CS à l'instant t.
- CS_m : représente le meilleur service concret sélectionné pour un service abstrait donné.

- Cp_m : est un compteur qui s'incrémente à chaque fois que l'exécution de service concret CSm réussit.
- $F(CSm)$: représente la mise à jour du facteur d'exploitation du service concret CSm dont l'exécution a réussi.

3) **Phase de vérification de la faisabilité du plan d'exécution** :

la phase précédent nous a permis de construire le plan d'exécution qui répond aux exigences locales de l'utilisateur. Compte tenu de l'importance de la contrainte tempotelle exigée par l'utilisateur, une phase de vérification de la faisabilité du plan d'exécution s'impose afin de vérifier que le temps de réponse global de la composition ne dépasse pas le délai maximal(QSeG) imposé par l'utilisateur.

L'estimation du temps de réponse considéré dans notre travail est basée sur une approche pessimiste qui considère les pires valeurs du temps de réponse pour chaque modèle de composition.

Pour estimer le temps de réponse global, l'ordre d'exécution des services concrets doit suivre l'un des modèles cités dans la littérature et résumés dans le tableau(3.2) :

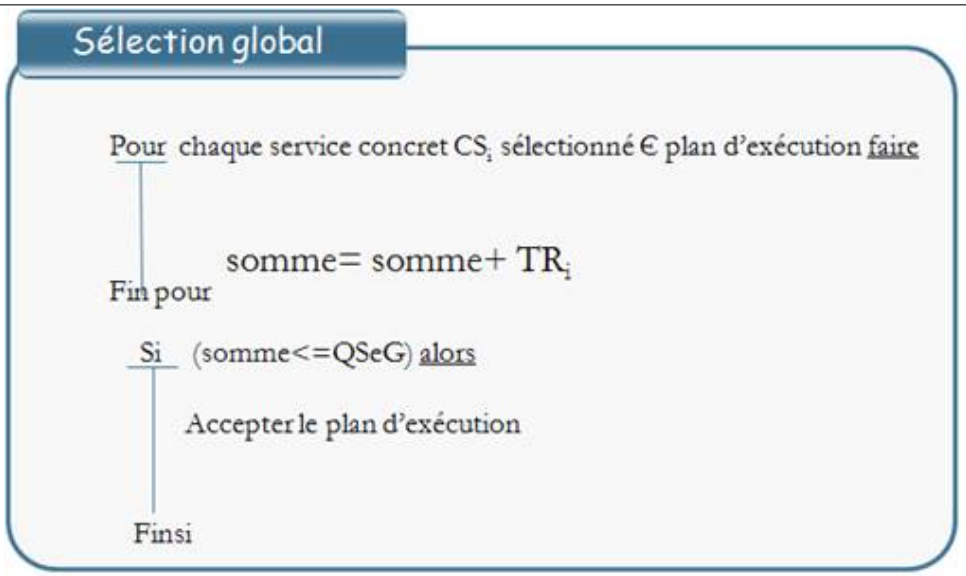
Attributs de QdS	Modèle de composition			
	SEQUENCE	ET	XOR	LOOP
Temps de réponse	n_i=1 RT_i	$Max(RT_i)$	$Max(RT_i)$	$RT_i * K$

TABLE 3.2 – Modèles de composion

- SEQUENCE : exécution séquentielle des services concrets ;
- ET : exécution parallèle des services concrets ;
- XOR : exécution conditionnelle des services concrets ;
- LOOP : exécution itérative des services concrets.

Dans le cadre de notre travail, nous avons pris le cas où l'exécution des services concrets se fait en séquentiel. Cela se justifie par le fait que les autres modèles peuvent être transformés en un modèle séquentiel [61].

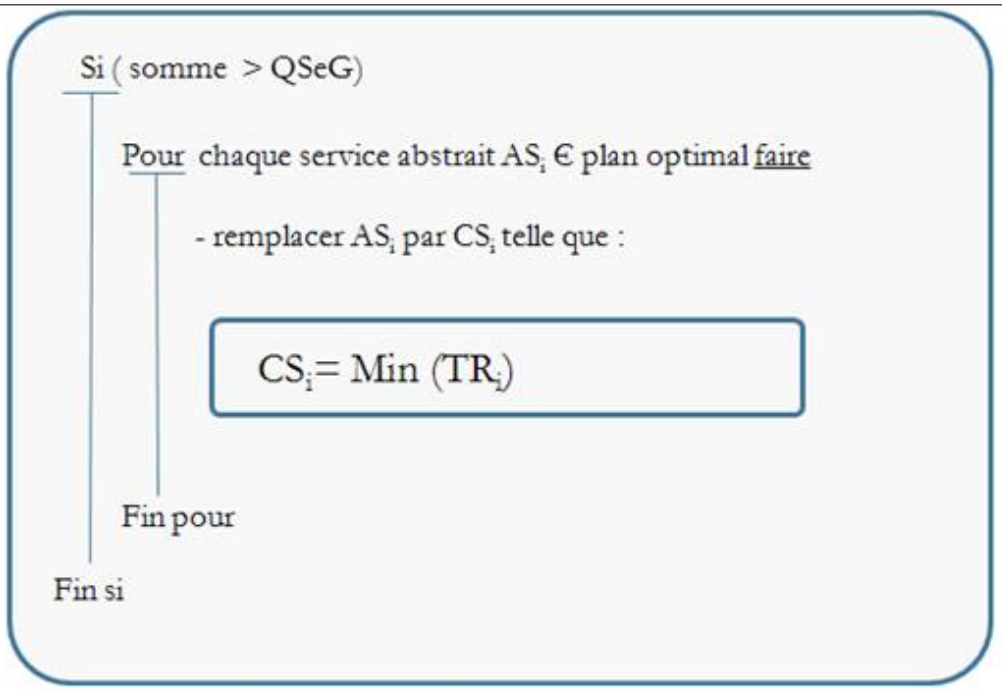
L'algorithme suivant illustre la phase de sélection global.



Où

- somme : représente la somme des temps de réponse des meilleurs services concrets du plan d'exécution.
- n : nombre de services concrets du plan d'exécution.
- QSeG : il représente la durée maximum que le temps d'exécution de l'ensemble des services combinés ne doit pas dépasser.

Dans le cas où le temps de réponse global dépasse la valeur du seuil QSeG, un autre plan d'exécution sera généré en substituant chaque service concret par un autre service de la même classe (service abstrait), qui minimise le temps de réponse. En fonction de ces nouvelles valeurs, la valeur de "somme" sera recalculée, puis on la compare avec le seuil QSeG. Dans le cas où somme > QSeG, l'exécution devient impossible.



3.6.2.4 Contrôle et exécution du plan concret

Après la sélection des services concrets les plus appropriés vérifiant les conditions exigées dans les deux phases de notre approche (présélection et sélection finale), le plan d'exécution sera exécuté par la suite selon un ordre établi.

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 le répertoire (CSD : *Concret Service Directory*) pour une utilisation ultérieure.

3.6.3 Schéma globale de l'approche proposée

Dans ce qui suit, nous décrivons les étapes de notre approche à travers un schéma globale.

Après la génération du plan global puis optimal conformément à la requête de l'utilisateur, nous avons obtenu le plan illustré dans la figure ci-dessous :

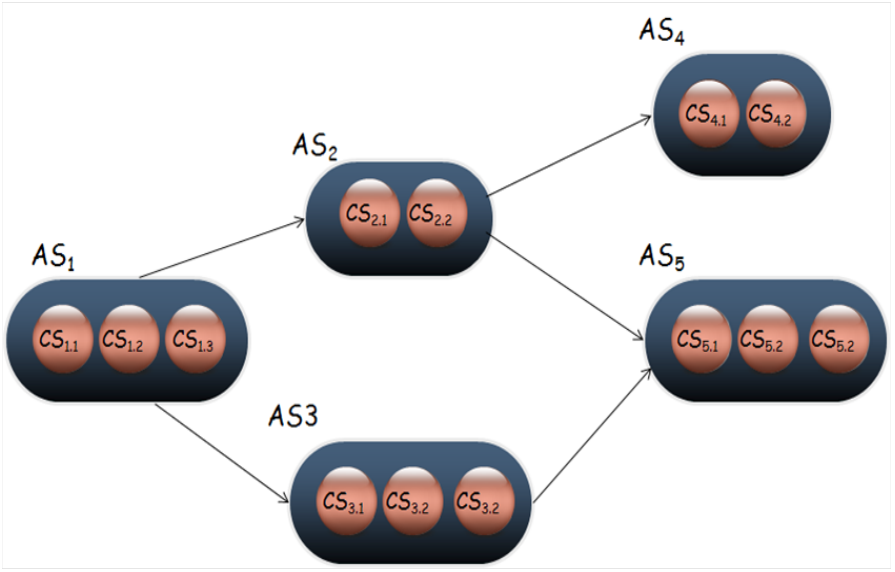


FIGURE 3.5 – Plan optimal généré.

Après la phase de mise à l'échelle, la première étape consiste à calculer la qualité d'expérience QdE de chaque service concret, et de sélectionner les services concrets qui dépassent le seuil local QSe exigé par l'utilisateur.

Le schéma ci-dessous illustre le résultat obtenu après la phase de présélection.

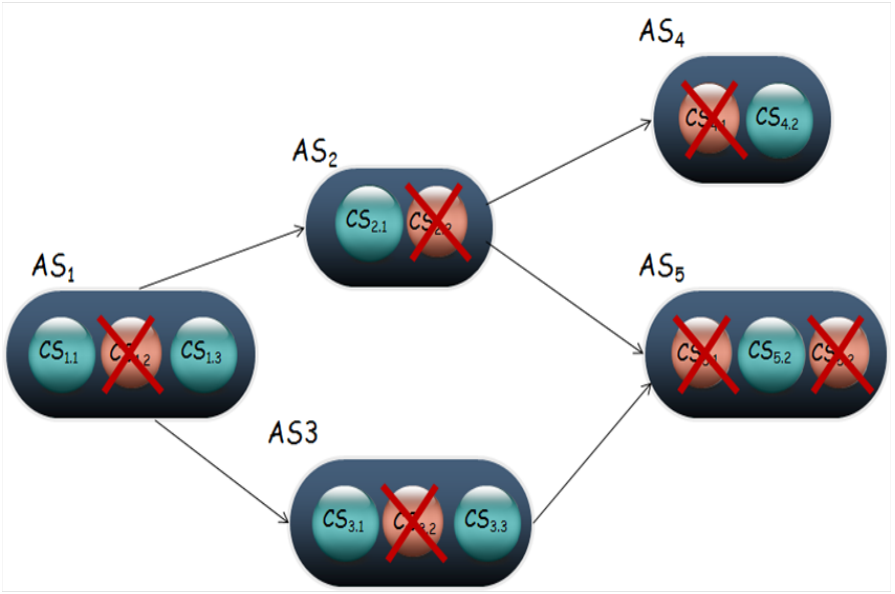


FIGURE 3.6 – Phase de présélection.

Une fois la qualité d'exploitation (notée QdEP) estimée en fonction du facteur d'exploitation (noté F) et de la qualité d'expérience (notée QdE) déjà calculée dans la phase de présélection, la fonction ArgMax() intervient pour trouver le meilleur service concret qui maximise la qualité d'exploitation pour chaque service abstrait. Ces services constituent le plan de composition qui pourrait

satisfaire la requête de l'utilisateur.

Le schéma ci-dessous illustre le résultat obtenu après la phase de sélection finale.

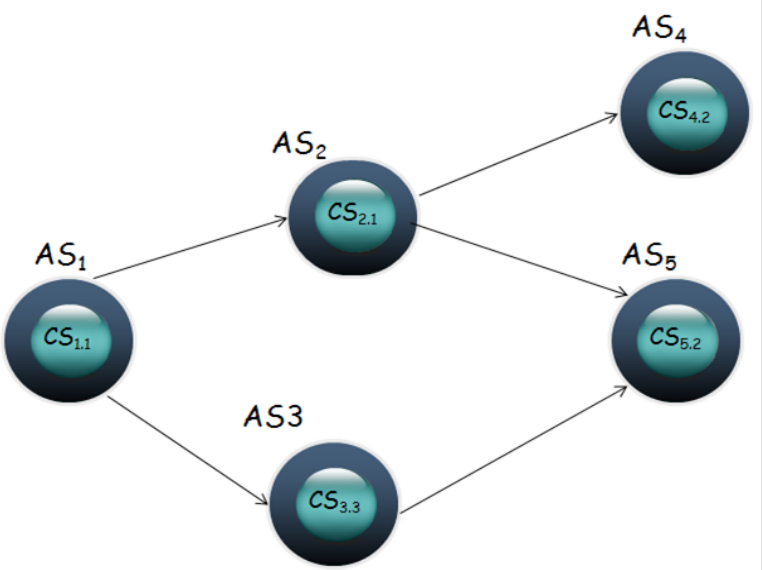


FIGURE 3.7 – Phase de sélection finale.

Après avoir généré le plan d'exécution, on vérifie sa faisabilité en calculant le temps de réponse du service composite (la somme des temps de réponse des services concrets qui le constituent), puis on compare ce temps avec le seuil global QSeG exigé par l'utilisateur. Dans notre exemple, la moyenne des temps de réponse de tous les services concrets sélectionnés est inférieure au seuil global QSeG, par conséquent, le plan d'exécution généré peut être exécuté.

3.6.4 Exemple d'application

Afin d'illustrer le déroulement de notre modèle de composition de services, nous utilisons un scénario pour l'assistance et la surveillance à domicile d'une personne dépendante [14].
Il s'agit particulièrement de surveiller les mouvements de la personne et de transmettre ensuite les informations aux prestataires de service (figure 3.6)

3.6.4.1 Le système de capture

Le domicile de la personne dépendante est muni d'un système de capture qui est représenté sur la figure suivante :

FIGURE 3.8 – Système de capteur du domicile de la personne dépendante.

Le réseau domestique est choisi comme exemple cible car il est un environnement pervasif par excellence.
Il est un terrain de prédilection pour l'expérimentation des modèles de conception recherchés.
Premièrement, la maison est un système ouvert : de nombreux utilisateurs et équipements divers peuvent aller et venir dans la maison et y séjourner plus ou moins longtemps. Ensuite, le réseau domestique est dynamique et distribué : l'utilisateur est mobile et vaque à de nombreuses activités pouvant être assistées par divers équipements qui se connectent, se déconnectent au gré de leurs déplacements dans la maison ou en raison de mesures de sauvegarde d'énergie.

Aussi, ces équipements distribués sont hétérogènes : leurs capacités, leurs langages de programmation, leurs protocoles de communication et leurs sémantiques d'utilisation sont variés. Enfin, les aspects embarqués sont exacerbés par la concurrence forte sur les prix de vente entre les fabricants de ce marché en plein essor.
Le tableau suivant résume les différents capteurs utilisés au domicile de la personne dépendante et leurs descriptions respectives.

Capteur	Désignation	Description
●	Analyse de l'environnement ambiante.	Température, humidité, luminosité
●	Présence (position)	Détecteur de position (peut être porté par l'utilisateur lui même)
●	Consommation d'eau	Détecteur de la consommation d'eau
●	Consommation électrique	Détecteur de la consommation électrique.
●	Caméra	Flux vidéo.
◯	Accéléromètre	Détecteur de mouvement
■	Ouverture porte / fenêtre	Ouverture des portes et fenêtres
■	Ouverture meuble	Manipulation des meubles
□	Activité cardiaque	Capteur de l'activité cardiaque de la personne (porté par la personne)
⊖	Robot mobile	Robot mobile équipé d'une caméra de surveillance

TABLE 3.3 – Exemple d'application : les capteurs et leur désignation

Les services abstraits disponibles dans cet environnement sont résumés dans le tableau(3.4) :

Service abstrait	Entrées	Sorties	Description
AS1	a	b	Service localisation
AS2	c	d	Analyse de l'environnement ambiant
AS3	e	f	Analyse du mouvement
AS4	g	h	Analyse de l'activité cardiaque
AS5	b, q	i	Surveillance des mouvements du robot
AS6	b, i	j	Service caméra
AS7	k	l	Mesure de la consommation électrique
AS8	m	n	Mesure de la consommation d'eau
AS9	o	p	Agenda des soins
AS10	F, h, p	q	Surveillance de l'état de la personne
AS11	q	r	Notification

TABLE 3.4 – Exemple d'application : les services abstraits.

Le tableau ci-dessous représente la description des entrées/sorties des services abstraits disponibles dans l'environnement domotique.

Paramètres	Désignation	Description
a	Signal position	Signal indiquant la position de la personne
b	Localisation	Signal de position
c	Vecteurs de donnés ambiantes	Température, luminosité, humidité.
d	Vecteur de données ambiantes analysées	Données analysées (exemple : température élevée)
e	Signal accéléromètre	Signal des mouvements de la personne
f	Analyse de mouvement	Mouvement interprété (exemple : personne tombée)
g	Signal cardiaque	Signal de l'activité cardiaque de la personne
h	Analyse cardiaque	Activité cardiaque analysée
i	Position robot	Position actuelle du robot
j	Flux vidéo	Flux vidéo transmis par la caméra
k	Vecteur de consommation électrique.	Consommation en électricité des différents dispositifs
l	Consommation en électricité.	Consommation totale en électricité.
m	Vecteur consommation eau	Consommation en eau des différents dispositifs
n	Total Consommation eau	Consommation totale en eau
o	Vecteur de soins	Historique des soins de la personne
p	Vecteur de soins analysé	Les soins prévus (exemple : visites)
q	Etat de la personne	Etat de santé de la personne (exemple : bon, grave)
r	Notification	Signal indiquant une situation

TABLE 3.5 – Exemple d'application : les paramètres d'entrées/sorties

3.6.4.2 Le système de composition de services

Notre système de composition de services reçoit les données à partir du système de capture, puis effectue la composition afin de sélectionner les services appropriés pour la situation de la personne et communiquer avec des prestataires de services externes pour une éventuelle intervention, la figure suivante décrit le scénario de la surveillance à domicile d'une personne dépendante.

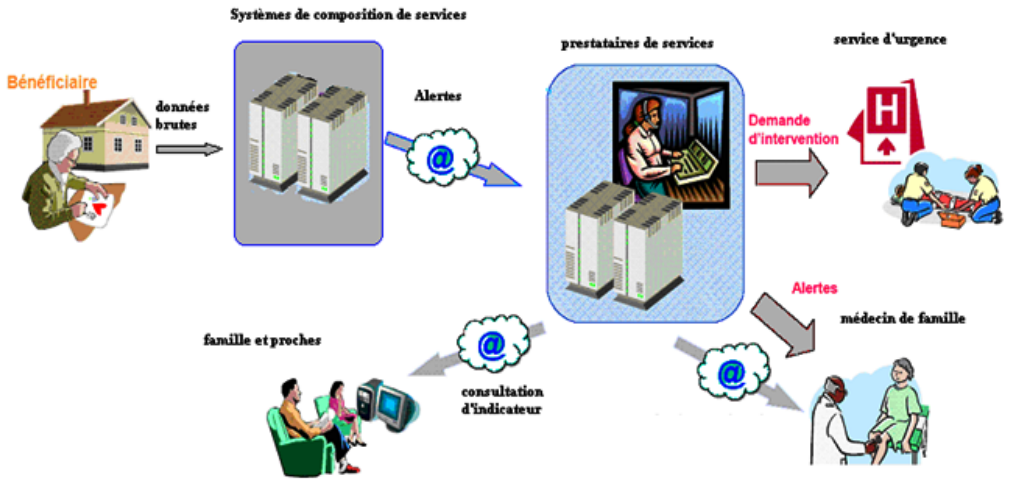


FIGURE 3.9 – Application du service de composition de services

Supposant qu'à un instant donné, le système de composition reçoit les paramètres (a, e, g, o) du système de capture et désire obtenir les paramètres (r, j) à transmettre pour les prestataires de services. L'application de l'algorithme de sélection par chainage arrière [49] des services abstraits donne les résultats suivants :

2) **Plan optimal abstrait de composition**

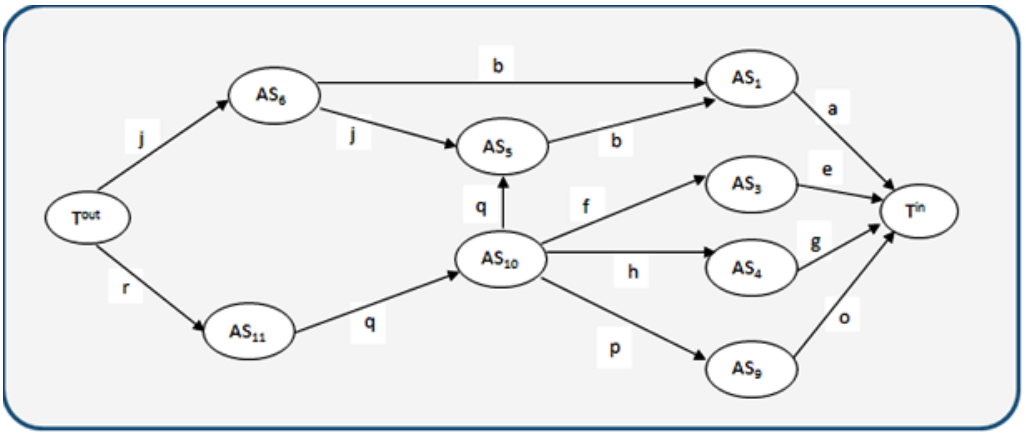


FIGURE 3.10 – Plan abstrait de composition de services

D'après la requête, exprimée implicitement par l'utilisateur, ayant comme entrés (a,e,g,o) = (signal de position, signal accéléromètre, Signal cardiaque, vecteur de soin), et comme sorties

(r,j) = (Notification, Flux vidéo). Trois paramètres de QdS apparaissent, auxquels l'utilisateur prête plus d'intérêt à savoir : la fiabilité, la disponibilité et le temps réel.

1. **La sélection des services concrets**
   Pour chaque service abstrait du plan optimal généré, nous devons sélectionner le meilleur service concret. L'ensemble de ces services constitue le plan concret d'exécution.

Pour illustrer l'élection du meilleur service concret, nous déroulerons notre approche sur l'ensemble des services abstraits du plan optimal. Pour le présent exemple, nous considérons que les services concrets de la même classe ont déjà participé à des exécutions ultérieures. Dans ce cas, nous appliquerons les règles de calculs et de mise à jour définies précedement.

Afin de trouver l'ensemble des services concrets à combiner pour répondre à la requête complexe précitée, nous supposons également que les préférences de l'utilisateur sont de 0,2 pour la fiabilité , de 0,4 pour la disponibilité , de 0,5 pour le temps réel , de 0,5 pour le seuil QSe et de 4,2 pour le seuil QSeG .

Apres avoir généré le plan abstrait, nous déroulons notre approche de composition sur l'ensemble des services abstraits sélectionnés, afin de trouver le meilleur service concret pour chaque service abstrait.

Après la phase de mise à l'échelle supposé être faite , la première étape consiste à calculer la qualité d'expérience QdE de chaque service concret, le tableau suivant résume les résultats obtenus :

ASi	**Services concrets**	**Temps de réponse**	**Fiabilité**	**Disponibilité**	**Sécurité**	**Temps réel**	**QdE**
AS1	SC1.1	0.5	0.6	0.2	1	0.92	0.64
	SC1.2	0.6	0.89	0.92	0	0.67	0.8
	SC1.3	0.89	0.56	0.48	1	0.39	0.47
AS3	SC3.1	0.56	0.98	0.23	0.66	0.56	0.66
	SC3.2	0.17	0.18	0.89	0.66	0.78	0.56
AS4	SC4.1	0.2	0.32	0.98	1	0.56	0.54
	SC4.2	0.65	0.8	0.20	0	0.23	0.45
AS5	SC5.1	0.23	0.51	0.92	0.33	0.21	0.47
	SC5.2	0.63	0.97	0.36	0.33	0.49	0.54
	SC5.3	0.25	0.67	0.89	1	0.25	0.54
AS6	SC6.1	0.68	0.98	0.39	0.33	0.19	0.55
	SC6.2	0.36	0.99	0.80	0	0.23	0.65
AS9	SC9.1	0.23	0.56	0.29	0.33	0.48	0.47
	SC9.2	0.56	0.65	0.86	0	0.56	0.56
	SC9.3	0.63	0.18	0.15	1	0.19	0.5
AS10	SC10.1	0.56	0.78	0.96	0.33	0.29	0.62
	SC10.2	0.60	0.29	0.33	0	0.95	0.56
AS11	SC11.1	0.26	0.72	0.69	0	0.96	0.81
	SC11.2	0.56	0.71	0.78	0.66	0.71	0.72
	SC11.3	0.56	0.56	0.3	0.66	0.6	0.52

TABLE 3.6 – Phase de présélection des services concrets

En utilisant les valeurs obtenues dans le tableau ci-dessus, nous sélectionnons dans chaque service abstrait, les services concrets dont leurs QdE dépassent le seuil QSe exprimé par l'utilisateur :

ASi	Services concrets	Temps de réponse	Fiabilité	Disponibilité	Sécurité	Temps réel	QdE	F	QdEP
AS1	SC1.1	0.5	0.6	0.2	1	0.92	0.64	0.4	0.25
	SC1.2	0.6	0.89	0.92	0	0.67	0.8	0.63	0.5
AS3	SC3.1	0.56	0.98	0.23	0.66	0.56	0.66	0.53	0.35
	SC3.2	0.17	0.18	0.89	0.66	0.78	0.56	0.72	0.40
AS4	SC4.1	0.2	0.32	0.98	1	0.56	0.54	0.65	0.35
	SC5.2	0.63	0.97	0.36	0.33	0.49	0.54	0.52	0.0.3
	SC5.3	0.25	0.67	0.89	1	0.25	0.54	0.82	0.43
AS6	SC6.1	0.68	0.98	0.39	0.33	0.19	0.55	0.52	0.28
	SC6.2	0.36	0.99	0.80	0	0.23	0.65	0.18	0.15
	SC9.2	0.56	0.65	0.86	0	0.56	0.65	0.45	0.29
AS10	SC10.1	0.56	0.78	0.96	0.33	0.29	0.62	0.65	0.40
	SC10.2	0.60	0.29	0.33	0	0.95	0.56	0.61	0.34
AS11	SC11.1	0.26	0.72	0.69	0	0.96	0.81	0.25	0.20
	SC11.2	0.56	0.71	0.78	0.66	0.71	0.72	0.51	0.36
	SC11.3	0.56	0.56	0.3	0.66	0.6	0.52	0.62	0.32

TABLE 3.7 – Phase de sélection finale des services concrets

Dans le tableau précédent, la qualité d'exploitation de chaque service concret est estimée en fonction de la qualité de l'expérience et le facteur d'exploitation. Le plan d'exécution est obtenu en choisissant dans chaque service abstrait le service concret qui maximise la qualité de l'exploitation. Le plan d'exécution obtenu est représenté dans la figure ci-dessous :

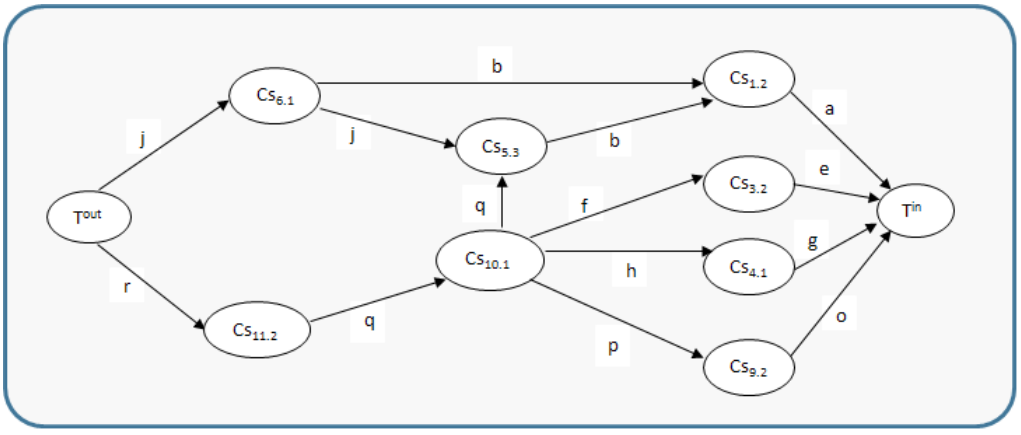


FIGURE 3.11 – Génération du plan concret.

Après avoir généré le plan d'exécution, on vérifie sa faisabilité en calculant le temps de réponse de tous les services concrets qui le constituent, puis on compare ce temps avec le seuil QSeG(global) exigé par l'utilisateur. Les résultats de la sélection globale est résumé dans le tableau suivant :

CSi	CS1.1	CS3.3	CS4.1	CS5.1	CS6.4	CS9.2	CS10.1	CS11.2
TRi	0,6	0,17	0,2	0,25	0,68	0,56	0,56	0,56
Somme	(0,6+0,17+0,2+0,25+0,68+0,56+0,56+0,56)= 3,58							

TABLE 3.8 – Sélection finale des services concrets

D'après le résultat du temps de réponse global calculé (Somme = 3,58), on remarque que cette valeur est inferieur au seuil global QSeG(4,2) exprimé par l'utilisateur.Par conséquent, le plan d'exécution généré peut être exécuté, car il remplit tous les critères escomptés et exigés par l'utilisateur.

3.7 Conclusion

En raison des progrés actuels des systèmes informatiques, il est tout naturel de prendre en compte le contexte de l'utilisateur afin de mieux le satisfaire. Dans ce contexte relationnel hautement dynamique, la notion de QdS, qui s'intéresse à la qualité de la relation entre un service et son client, devient un enjeu crucial.

A partir de ce constat, nous avons proposé un modèle pour la composition automatique qui satisfait les locales et globales de l'utilisateur en sélectionnant les meilleurs services concrets selon les paramètres de QdS prise en compte. Afin de montrer l'application concrète de notre approche pour la composition dynamique de services, nous avons étudié un cas réel qui porte sur les services pour l'assistance et la surveillance d'une personne dépendante.

Dans le prochain chapitre, nous évaluerons les performances de notre approche et nous montrerons également les résultats des tests d'implémentation.

CHAPITRE 4

SIMULATION ET ANALYSE DE PERFORMANCES DE L'APPROCHE PROPOSÉE

4.1 Introduction

L'approche de composition dynamique de services avec QdS dans les environnements ubiquitaires développée dans le chapitre précédant est le sujet d'implémentation et de validation du présent chapitre.

En premier, nous expliquons la démarche que nous avons adoptée pour valider notre contribution et nous présentons les étapes principales suivies. Le second volet met l'accent sur la validation par simulation de la solution proposée et l'évaluation de ses performances, et ce en la comparant avec l'une des approches proposées [52] dans la littérature.

4.2 Le but de l'implémentation

Le but de notre travail étant de développer un simulateur afin de valider notre proposition, d'en évaluer les performances, et d'en mesurer le réel apport en terme de temps d'exécution.
La simulation est le fait de reproduire le comportement dans le temps d'un phénomène réel, cela permet d'observer l'évolution d'un système en faisant varier certains paramètres affectant son comportement. La simulation permet de pousser plus loin des tests qui ne sont pas toujours réalisables à cause de contraintes de ressources (temps, coût du matériel qu'on souhaite tester, etc...).

Nous évaluons aussi l'efficacité et la rapidité de l'algorithme de génération du plan concret et le rôle de la phase de présélection dans la réduction du temps de réponse. Tous ces tests ont été exécutés sur un PC Core 2 Duo avec 2 Go RAM, processeur de 2.1 GHZ sous windows.

Le but de ces tests est d'évaluer également la scalabilité de notre approche, c.à.d. la possibilité de générer des plans concrets pour un nombre très grand de services abstraits ou concrets.

4.3 Le choix du langage utilisé

Notre système de composition de services a été implémenté en JAVA sous la plateforme Eclipse version 3.5.0. en effet, Eclipse est un environnement de développement intégré (Integrated Development Environment) dont le but est de fournir une plateforme modulaire pour permettre de réaliser des développements informatiques. Il est spécialement conçu pour le langage de programmation Java. Grâce à son interface complète et accessible, Eclipse permet de développer des sites Web et logiciels en toute simplicité. En plus d'être doté de nombreuses performances, Eclipse est gratuit et open-source ce qui fait de lui un concurrent de taille face aux autres environnements de développement.

4.4 Présentation du système de composition de services

Notre système représente un ensemble de services concrets qui sont mis à disposition pour une éventuelle utilisation dans un répertoire des services concrets (CS). Ces services sont classés en fonction de leurs entrés et sorties dans des services abstraits (AS).

A partir de la requête de l'utilisateur, on cherche à trouver un service concret pouvant répondre à cette requête ; si aucun service atomique n'est trouvé, on fait appel au système de composition permettant de sélectionner les services concrets à composer afin de répondre à la requête de l'utilisateur.

Dans nos expériences, nous avons varié le nombre de services abstraits ainsi que le nombre de services concrets de chaque service abstrait à chaque invocation. Nous avons également généré aléatoirement les préférences de l'utilisateur associées à sa requête. Nous avons par la suite mesuré les performances de notre approche et celles de l'approche CDSC proposée par Tari et al. [52] en terme de " temps de génération du plan concret ". En effet, ce dernier est l'un des principaux critères de performances pour les méthodes de composition, il représente le temps nécessaire pour la construction du plan du service composite à partir du plan abstrait généré. Cette métrique est affectée par des facteurs comme le nombre de services dans les scénarios de simulation, la présence de défaillances et le nombre de services abstraits.

4.5 Variables descriptives du système

Les variables utilisées dans notre système sont illustrées dans le tableau4.1 :

Définition de la variable	Nom de la variable	Type	Valeur initiale
Qualité d'expérience	Qoe	réel	0
Qualité d'exploitation	QoEP	réel	0
Le facteur d'exploitation	Facteur	réel	1
Le répertoire services concret	CS	vecteur	//
Le répertoire de services abstraits	ASD	Vecteur	//
Un vecteur contenant les services abstraits du plan abstrait généré.	planAS	vecteur	//
Un répertoire contenant les meilleurs services concrets sélectionnés.	execution	vecteur	Les éléments de ce vecteur sont initialisés à 0.
Les préférences de l'utilisateur.	pref	vecteur	Les éléments de ce vecteur sont initialisés à 0.
Le seuil local	QSe	Réel	//
Le seuil global	QSeG	Réel	//

TABLE 4.1 – Variables descriptives du système.

4.6 Etape de réalisation

Pour implémenter notre approche de composition de services, nous avons suivi les étapes décrites dans la figure (4.1) :

FIGURE 4.1 – Etapes de réalisation du simulateur.

– **Initialisation des variables :**cette étape consiste à générer les entrées, les sorties, la réputation (initialisée à 1) et un nombre de services concrets pour chaque service abstrait constituant le plan. Nous avons également initialisé les paramètres de qualité de services de chaque service concret.

– **Génération des préférences de l'utilisateur :** cette étape permet à l'utilisateur d'introduire ses préférences.

– **Estimation de la qualité d'expérience :** dans cette étape, la qualité d'expérience QdE de chaque service concret est estimée en fonction des préférences de l'utilisateur. En effet, la QdE est une somme pondérée des paramètres de qualité de service (QdS). Le poids associé à chaque paramètre de QoS est mesuré en fonction des préférences de l'utilisateur (indiqué dans son profil), son contexte, et le contexte de l'environnement.

– **La présélection des services concrets selon le seuil QSe :** : elle consiste à sélectionner pour chaque service abstrait, les services concrets dont la qualité QdE dépasse le seuil QSe exigé par l'utilisateur.

– **Estimation de la qualité d'exploitation :** pour chaque service concret sélectionné dans la phase de présélection, la qualité de l'exploitation QdEP est mesurée en fonction de la qualité

d'expérience (QdE) et du facteur d'exploitation (F).

- **Génération du plan concret :** il consiste à sélectionner pour chaque service abstrait le service concret qui maximise sa qualité d'exploitation QdEP.

- **Vérification de la faisabilité du plan d'exécution :** il consiste à vérifier que le temps de réponse du service composite ne dépasse pas le délai QSeG exigé par l'utilisateur qui représente le temps maximal que l'exécution du plan concret ne doit impérativement pas dépassé.

- **Exécution et mise à jour des variables des services concrets du plan concret :** cette étape consiste à générer aléatoirement des variables représentant l'état de l'exécution des services concrets. En fonction de cet état, des mises à jour des paramètres de QdS des services concrets sont prévues à cet égard.

4.7 Evaluation des performances

Nous avons mesuré la variation du temps de génération du plan concret de notre approche (QDSC) en fonction du nombre de services concrets. La comparaison des résultats a été faite avec ceux estimés dans l'approche CDSC [52]. Pour cela, nous avons fixé le nombre de services abstraits à 5 puis à 10, et nous avons varié le nombre de services concrets de 100 à 1000. Pour obtenir le temps moyen de génération du plan concret, 10 simulations ont été exécutées.

Les simulations ont été effectuées sur des services concrets avec une génération aléatoire des services défaillants lorsque notre programme s'est exécuté en écartant la phase de sélection globale, cela a été suivi par phase de replanification afin de remplacer le service dont l'exécution a échoué.

Les figures(4.2, 4.3) représentent, pour 5 et 10 services abstraits respectivement, la variation du temps de génération du plan concret en fonction du nombre de services concrets.

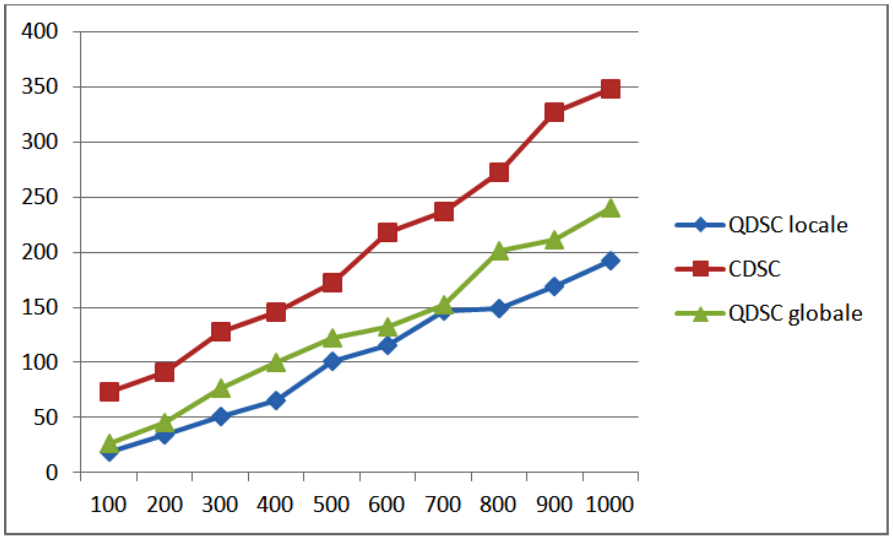


FIGURE 4.2 – Temps de génération du plan concret avec 5 services abstraits.

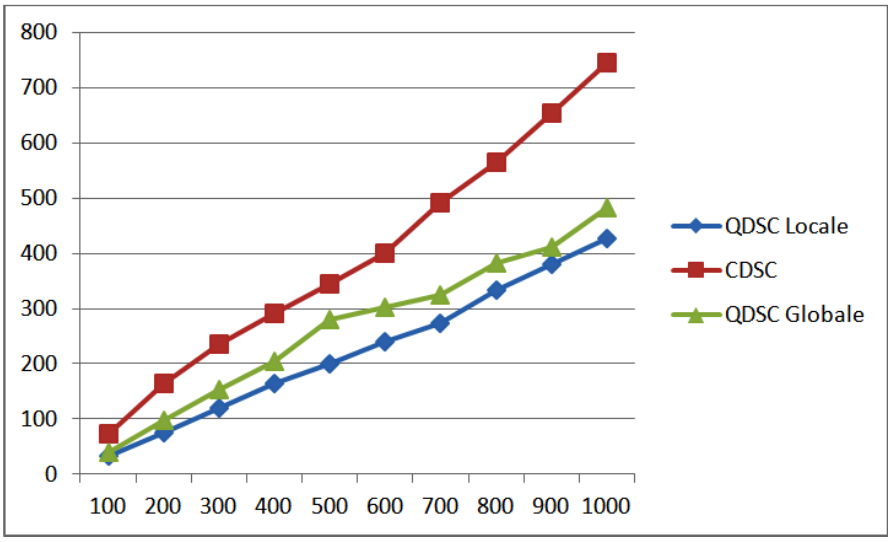


FIGURE 4.3 – Temps de génération du plan concret avec 5 services abstraits.

Les résultats montrent que notre proposition est meilleure que l'algorithme CDSC qui ne prend pas en charge que la sélection locale. Ce gain peut atteindre les 300 ms pour la sélection locale et 250 ms pour la sélection globale (pour 1000 services concrets par exemple) par rapport à l'approche CDSC. Cet apport est dû à la phase de présélection qui contrairement à l'algorithme CDSC, sélectionne tout d'abord les services concrets qui satisfont les exigences locales (dépassent le seuil QSe) exigé par l'utilisateur, et seuls ces services seront comparés grâce à la fonction ArgMax() pour choisir le meilleur service concret qui fera partie du plan d'exécution.

Il est important de noter que le temps de la sélection globale superforme le temps de la sélection locale, étant donnée les tests supplémentaires effectuées avant l'exécution du plan concret dans le cas de la sélection globale.

Egalement, sur les mêmes figures, on peut remarquer que l'accroissement du nombre de services concrets déployés dans l'environnement induit à une augmentation dans le temps de génération des deux solutions. Ceci est attendu car le fonctionnement de ces deux approches dépend fortement du nombre de services déployés dans l'environnement.

4.8 Conclusion

Ce chapitre a fait l'objet d'une comparaison des performances de notre proposition avec l'approche [52] à l'aide d'un simulateur réalisé en JAVA sous la plateforme Eclipse. Les tests réalisés ont mené à étudier des différentes caractéristiques de notre solution et d'évaluer ses performances en terme de résultats produits et de temps d'exécution du plan concret.

Les résultats des tests montrent que notre approche améliore le temps de génération du plan concret, cela est principalement dû à l'amélioration portée lors de la phase de présélection des services concrets du plan abstrait généré.

CONCLUSION ET PERSPECTIVES

La vision de l'informatique diffuse telle qu'introduite par Weiser devient une réalité grâce aux progrès réalisés ces dernières années dans le domaine de la miniaturisation de composants électroniques, au faible coût des objets communicants et à l'émergence ainsi que l'expansion des technologies réseaux sans fil. L'objectif de l'informatique diffuse est de permettre aux utilisateurs d'accéder à des ressources et/ou à des services n'importe où et à tout instant à travers divers objets communicants. Pour réaliser cette vision, ces objets doivent être en mesure de découvrir les services de leur environnement immédiat et d'interagir entre eux afin de répondre au besoin de l'utilisateur.

Ce travail est le résultat de notre réflexion sur la problématique de la composition de services en environnement pervasif. 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 également d'adresser des mécanismes de sélection des meilleurs services concrets en termes de la QdS et de sensibilité au contexte. Ces mécanismes doivent permettre une adaptation continue à des situations changeantes, et doivent aussi tenir compte des incertitudes sur les informations disponibles pour maintenir un équilibre entre l'automatisation du fonctionnement et le contrôle de l'utilisateur. Nous avons intégré le facteur d'exploitation au processus de sélection afin de mesurer le degré de réponse de chaque service concret à chaque invocation et pouvoir ainsi choisir pour la composition les meilleurs en fonction du facteur et de la qualité d'expérience (QdE).

La nature dynamique des environnements dans lesquels les systèmes s'exécutent soulève des défis majeurs, notamment la création et le maintien de ces systèmes qui pourraient être critiques et complexes. Le temps réel permet de prendre en compte le moment ou le résultat est fourni dans les systèmes au fonctionnement critique. Si les contraintes temporelles ne sont pas satisfaites, on dit qu'une défaillance s'est produite. A partir de ce constat, nous avons introduit le temps réel comme paramètre de qualité de service (QdS) permettant de gérer l'aspect critique des services qui participeront au processus de composition.

Afin de valider notre approche (Quality-aware Dynamic Service Composition), nous avons implémenté notre approche sous Java dans le but de mesurer sa scalabilité et son apport en terme de temps de composition. Dans tous les tests de simulations effectués, les résultats indiquent que notre proposition est meilleure que l'approche CDSC, et ce pour la sélection locale et globale.

En guise de perspectives, nous envisageons de considérer la gestion des conflits lorsque plusieurs

utilisateurs cohabitent dans un même environnement lors du processus de composition de services, ainsi que la construction d'un environnement intelligent, en se basant sur un simulateur approprié tel que USARsim, afin de pousser plus loin nos tests à travers un scénario réel. Nous visons également prendre en compte la dégradation des paramètres de QdS lors de l'exécution du service composite, et d'étendre la phase de sélection globale, non seulement au temps de réponse, mais aussi à d'autres paramètres de qualité de service.

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 Toulouse 1 Capitole, France, décembre 2010.

[2] M. MIRAOUI. Architecture logicielle pour l'informatique diffuse : Modélisation du contexte et adaptation dynamique des services. Thèse de doctorat en Génie, Ecole de technologie supérieure université du Québec, Canada, juillet 2009.

[3] A. BOTTARO. Home SOA : Composition contextuelle de services dans les réseaux d'équipements pervasifs. Thèse de doctorat en informatique, Université Joseph Fourier-Grenoble, France, décembre 2008.

[4] Y. BROMBERG. Résolution de l'hétérogénéité des intergiciels d'un environnement ubiquitaire. Thèse de doctorat en informatique, Université de Versailles Saint-Quentin-En-Yvelynes, France, 2006.

[5] Amel KOUICEM.Composition dynamique de services en environnements ubiquitaires. Mémoire de magister en informatique option : réseaux et systèmes distribués de l'université de Bejaia, 2008.

[6] A. CHIBANI. Intergiciel multi agents orienté web sémantique pour le développement d'applications ubiquitaires sensibles au contexte. Thèse de doctorat en Informatique-Intelligence Artificielle, Université Paris XII-Val de Marne, France, décembre 2006.

[7] F. SEBBAK. Architecture intergicielle sémantique basé sur le standard OSGi pour les services robotiques. Mémoire de Magistère en Informatique, Université Abderrahmane Mira de Bejaïa, Algérie, 2006/2007.

[8] C. TACONET. Intergiciels pour la sensibilité au contexte en environnement ubiquitaire. Université d'Evry-Val-d'Essonne. Thèse doctorat, 2011.

[9] A. CARRILLO RAMOS. Agents ubiquitaires pour un accès adapté aux systèmes d'information : Le Framework PUMAS. Thèse de doctorat en informatique, Université Joseph Fourier, France, mars 2007.

[10] K. TIEU. Mobilité, Contexte et Adaptation dans un système informatique éducatif. Mémoire de fin d'étude Master en informatique, École Nationale Supérieure des Télécommunication de Bretagne, France, août 2007.

[11] C.DUMEZ. Approche dirigée par les modèles pour la spécification, la vérification formelle et la mise en œuvre de services Web composés. Thèse de doctorat en Informatique, Université de Technologie de Belfort-Montbéliard, France, août 2010.

[12] R. Nzekwa. Support intergiciel pour l'auto-adaptation stable dans les environnements ubiquitaires. Thèse de Master en Informatique de l'Université des Sciences et Technologies de Lille, France, Juin 2009.

[13] S.HAMMACHI, R.HOUARI. La protection des données privées dans les systèmes ubiquitaires. Thèse de Master en Informatique, Université Abderrahmane Mira de Bejaïa, Algérie, Juin 2010.

[14] A.YACHIR. Modélisation et composition de services sensibles au contexte. Thèse de doctorat en Informatique, Université Abderrahmane Mira de Bejaïa, Algérie, 2006/2007

[15] OWL : http ://www.w3.org/TR/webont-req/.12/06/2012.

[16] Rahwan, T., Rahwan, T., Rahwan, I., Ashri, R. Agent-Based Support for Mobile Users Using AgentSpeak (L).Chicago, USA, October 13, 2003.

[17] Hristova, N., O'Hare, G. Ad-me : wireless advertising adapted to the user location, device and emotions.part of the Software Technology Track, IEEE Computer Society Press (2004), pp. 1-10.

[18] B. Schilit et M. Theimer : Disseminating Active Map Information to Mobile Hosts. IEEE Network, 8(5) :22-32, 1994.

[19] N. S. Ryan, J. Pascoe et D. R. Morse : Enhanced Reality Fieldwork : the Context-aware Archaeological Assistant.In V. Gaffney, M. van Leusen et S. Exxon, éditeurs : Computer Applications in Archaeology 1997, British Archaeological Reports, Oxford, October 1998. Tempus Reparatum.

[20] J. Pascoe, Adding generic contextual capabilities to wearable computers. In Wearable Computers, 1998. Digest of Papers. Second International Symposium on, pages 92-99, octobre 1998.

[21] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith et P. Steggles, Towards a Better Understanding of Context and Context-Awareness. In HUC '99 : Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing, pages 304-307, London, UK, 1999. Springer-Verlag.

[22] Chen, G., et D. Kotz. 2000, A Survey of Context-Aware Mobile Computing Research. TR2000-381. Dartmouth : Dartmouth College Computer Science.

[23] Dey, A. K. 2001, Understanding and using context. Personal and ubiquitous computing vol. 5, p. 4-7.

[24] Brezillon, P., M. R. Borges, J.A. Pino et J.-Ch. Pomerol. 2004, Context-Awareness in Group Work : Three Case Studies. In 2004 IFIP Int. Conf. on Decision Support Systems (DSS 2004) (Juillet, 2004). Prato, Italie.

[25] B. Schilit, N. Adams et R. Want, Context-aware computing applications. In Mobile Computing Systems and Applications, WMCSA 1994. First Workshop on, pages 85-90, décembre 1994.

[26] N. S. Ryan, J. Pascoe et D. R. Morse : Enhanced Reality Fieldwork : the Context-aware Archaeological Assistant. In V. Gaffney, M. van Leusen et S. Exxon, éditeurs : Computer Applications in Archaeology 1997, British Archaeological Reports, Oxford, October 1998. Tempus Reparatum.

[27] G. Chen et D. Kotz : A survey of context-aware mobile computing research. Rapport technique, Dept. of Computer Science, Dartmouth College, 2000.

[28] F.BALIGAND.Une Approche Déclarative pour la Gestion de la Qualité de Service dans les Compositions de Services, thèse de doctorat en informatique, l'Ecole des Mines de Paris. juin 2008.

[29] M. Mrissa. Médiation Sémantique Orientée Contexte pour la Composition deServices Web, thèse de doctorat en informatique, Université Claude Bernard Lyon I. novembre 2007.

[30] Y. CHARIF. Chorégraphie dynamique de services basée sur lacoordination d'agents introspectifs, Thèse de doctorat en informatique, université pierre et Marie Curie Paris vi décembre 2007.

[31] M.DRISS. Approche multi perspectives centrée exigences de composition de services web, thèse de doctorat en informatique, université de rennes 1.decembre 2011.

[32] I.BRAKNI. Planification multi-agents pour la composition dynamique, thèseIngénioraten informatique, Université de Tébessa, 2010.

[33] T. HANH. Coordination adaptative de services à base de Contrats, thèse de doctorat en informatique, institut polytechnique de Grenoble, 2009.

[34] A.JMAL. Évaluation de Politiques d'Auto-Adaptabilité basées sur la Mobilité des Services Web Orchestré, Master en informatique, École Nationale d'Ingénieurs de Sfax, Tunisie, juillet 2009.

[35] D. BARREIRO CLARO. SPOC - Un canevas pour la composition automatique de services web dédiées à la réalisation de devis, thèse de doctorat en informatique, université d'Angers. 2006.

[36] J. Rao and X. Su. A survey of automated web service composition methods. In Semantic Web Services and Web Process Composition, First International Workshop, SWSWPC 2004, volume 3387 of Lecture Notes in Computer Science, pages 43-54, San Diego, USA, 2004. Springer.

[37] S. Kouadri Mostéfaoui. Towards a Context-Oriented Services Discovery and Composition Framework. 2003.

[38] J. Gortmaker, M. Janssen, and R. W. Wagenaar. The advantages of web service orchestration in perspective. Proceedings of the 6th international conference on electronic commerce (icec'04) 2004, pp. 506515.

[39] G.GARDARIN. Xml, des bases de données aux services web, Dunod, 2007.

[40] G. PEDRAZA FERREIRAFOCAS : un canevas extensible pour la construction d'applications orientées procédé, thèse de doctorat en informatique, université Joseph fourrier Grenoble 1, Novembre 2009.

[41] M. Ghallab, D. Nau, et P. Traverso, Automated Planning : Theory and Practice, edition Morgan Kaufmann, 2004.

[42] R. Reiter. Knowledge in Action : Logical Foundations for Specifying and Implementing Dynamical Systems.The MIT Press, 2001.

[43] P. Regnier, Algorithmique de la Planification en Intelligence Artificielle, édition Cepadues, 2004.

[44] H. Levesques, F. Pirri, and R. Reiter. Foundations for the situation calculus. Electronic Transactions on Artificial Intelligence, 2(3-4) :159-178, 1998.

[45] M.Vallée, L.Vercouter, F.Ramparany : Composition flexible de services d'objets communicants pour la communication ambiante, 2006

[46] J. Ferber, Les Systèmes Multi agents. Vers une intelligence collective. Interéditions,1995.

[47] J. Ferber. Les systèmes multi-agents. Vers une intelligence collective. Inter-Editions,1995.

[48] A. Yachir1, K. Tari, A. Chibani and Y. Amirat.Towards an Automatic Approach for Ubiquitous Robotic Services Composition.2008

[49] K. Tari , Y. Amirat, A. Chibani, A. Yachir : Rule-based Approach for Automatic Service Composition in Ubiquitous Environment. The 6th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI 2009).

[50] S. Sanlaville, Environnement de procédé extensible pour l'orchestration : Application aux services web, Ph.D. Thesis, 2005.

[51] A. Yachir1, K. Tari, Y. Amirat and A. Chibani.QoS Based Framework for Ubiquitous Robotic Services Composition, 2009

[52] K. Tari, Y. Amirat, A. Chibani, A. Yachir, A. Mellouk. Context-awareDynamic Service Composition in Ubiquitous Environment.2010.

[53] G. K. Mostéfaoui, G. Pasquier-Rocha, and P. Brézillon. Context-aware computing : A guide for the pervasive computing community. In ICPS, pages 39-48, 2004.

[54] N. Arenaza. Composition semi-automatique de services web. Thèse de Master. Ecole polytechnique de Lausanne. Février 2006.

[55] Yves Demazeau. Principes et architecture des systèmes multi agents, 2001.

[56] M. Wooldridge. An Introduction to Multiagent Systems, 340 p, John Wiley & Sons Publishers, Chichester, England, 2002.

[57] N. Ben Mabrouk, S. Beauche, E. Kuznetsova, N. Georgantas, V. Issarny. QoS-aware Service Composition in Dynamic Service Oriented Environment. INRIA Paris-Rocquencourt, France.Published in "MIDDLEWARE (2009)" DOI : 10.1007/978-3-642-10445-9_7.

[58] R. BEN Halima. Conception, implantation et expérimentation d'une architecture en bus pour l'autoréparation des applications distribuées à base de services web. Thèse de doctorat. Université de Toulouse. Mai 2009.

[59] M. Liu, M.Wang, W. Shen, N.Luo, J.Yan. A quality of service (QoS)-aware execution plan selection approach for a service composition process. Future generation computer system 28(2012).2011.08.017.

[60] R. KOCIK. Optimisation des systèmes distribués temps réel embarqués : application au prototypage rapide d'un véhicule électrique autonome. Thèse de doctorat en informatique industrielle, Université de ROUEN U.F.R de Sciences, France, mars 2000.

[61] M.Alrifai, T.Risse. Combining Global Optimisation with Local Selection for Efficient QoS-aware Service Composition.2009

[62] M.N. Huhns and M.P. Singh, editors. Readings in Agents. MOrgan Kaufmann, San Francisco, CA, USA, 1998.

[63] N.R. Jennings. An Agent-based Approach for Building Complex Software Systems.Communication of the ACM, 44(4) :35-41, 2001.

[64] Pascal Roques. UML2 par la pratique, 5ème édition.