République Algérienne Démocratique et Populaire
Ministère de l'enseignement supérieur et de la recherche scientifique

Université Abderrahmane Mira- Béjaia
Faculté des sciences exactes
Département d'informatique

Université Montpellier 2
Sciences et technologies.

Mémoire de fin de cycle

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

Option :

Systèmes Complexes, Technologies de l'Information et du Contrôle

Sélection globale de services en informatique ubiquitaire

Présenté par

Kamel MADI

Devant le jury composé de :

Président : Abdelkamel TARI	MCA	Université de Bejaïa, Algérie
Rapporteur : Yacine AMIRAT	Professeur	Université Paris Est Créteil
Examinateur : M. Essaid KHANOUCHE	MAB	Université de Bejaïa, Algérie
Examinateur : Zoubeyr FARAH	MAA	Université de Bejaïa, Algérie

2011-2012

Dédicaces

Ce modeste travail est dédié

A mes chers parents

A mes frères et mes sœurs

A toute ma famille

A mes amis(es).

Remerciements

Mes vifs remerciements vont d'emblée à Dieu le tout puissant pour son aide gracieuse.

Je tiens à remercier chaleureusement le Pr. Yacine Amirat d'avoir accepté de diriger mes travaux.

Un merci tout particulier au Dr. Tari AbdelKamel et au Dr Abdelghani Chibani pour leur aide et leurs conseils précieux.

Je remercie fortement tous mes enseignants de l'université de Béjaia, de l'université de Montpellier II et de l'université d'Aix Marseille II, qui ont appuyé mon cursus universitaire.

Un grand merci aux responsables du programme d'excellence « Averroès » à l'université de Montpellier et à l'université de Bejaia.

J'exprime ma profonde reconnaissance et mes vifs remerciements à Mr Ali Yachir pour m'avoir encadré et accordé son aide précieuse tout au long de ce stage.

Mes remerciements s'adressent aussi à Messieurs Abdel Kamel Tari, Achour Achroufene, et Khanouche M. Essaid, pour l'intérêt qu'ils ont bien voulu porter à ce travail en acceptant de le juger.

J'adresse également mes plus sincères remerciements à mes très chers parents ainsi qu'à toute ma famille ; sans eux rien n'aurait pu être possible.

Un grand merci à mes ami(e)s et mes collègues du Lissi, de l'université de Montpelier 2 et de l'université de Béjaia qui m'ont toujours soutenu et apporté un grand support moral.

Résumé

L'informatique ubiquitaire est devenu aujourd'hui une réalité grâce à la mise en réseau d'un nombre croissant de dispositifs informatiques par le biais de technologies réseaux sans fil. Dans ce mémoire nous proposons une approche de sélection globale de services, basée sur le formalisme CSP. Cette approche permet, d'une part, d'effectuer une sélection locale en prenant en considération les préférences et les exigences locales de l'utilisateur et d'autre part, de vérifier les exigences globales et de choisir le service composite optimal en termes de qualité de service. Cette approche prend en considération différents modèles du plan abstrait : séquentiel, parallèle, boucle et hybride. Elle prend également en compte plusieurs paramètres de qualité positifs et négatifs. Cette approche permet aussi une sélection dynamique des services en prenant en considération les pannes qui peuvent se produire lors de l'invocation des différents services sélectionnés.

La sélection, utilise par ailleurs les informations de contexte, comme le niveau d'énergie des équipements fournissant les services.

Mots clés : Informatique ubiquitaire, approche à services, QoS, Sélection globale, sélection locale, composition de service, CSP, heuristiques, contexte.

Abstract

Ubiquitous computing has become a reality thanks to the networking of a growing number of computing devices through wireless technologies. In this paper we propose a global selection of services, based on the CSP formalism. This approach allows, first, a local selection by taking into account local preferences and local requirements of the user and secondly, to check global requirements and select the optimal composite service in terms of quality of service. This approach takes into account different models of abstract plan: sequential, parallel, loop, and hybrid. It also takes into account several quality parameters positive and negative. This approach also allows dynamic selection of services taking into consideration the failures that may occur during the invocation of different services selected.

The selection also uses the context information, such as energy level of the equipment providing the services.

Keywords: Ubiquitous computing, Service oriented approach, QoS, Global selection, Locale selection, service composition, CSP, heuristic, Context.

Sommaire

Chapitre IV : *Implémentation et évaluatios des performances de l'approche proposée*

Liste des figures

Liste des figures

Introduction générale

En 1991, Mark Weiser présentait sa vision de l'informatique du 21eme siècle en établissant les fondements de ce que nous appelons aujourd'hui l'informatique ubiquitaire (*Ubiquitous Computing*). L'émergence de l'informatique ubiquitaire a été rendue possible grâce aux progrès réalisés ces dernières années dans la miniaturisation de composants électroniques et à l'émergence des technologies réseaux sans fil. Ainsi un nombre croissant d'objets de notre quotidien intègre désormais des dispositifs électroniques grâce auxquels ils deviennent communicants. Ces *objets communicants* peuvent être des dispositifs informatiques, des périphériques, ou des équipements de domaines aussi variés que l'électronique, l'électroménager, les télécommunications, la domotique ou l'automobile. Ces objets sont, selon le cas, soit mobiles ou immobiles.

L'approche à services traite directement la vision de l'informatique ubiquitaire. L'émergence de cette approche a été accompagnée par une augmentation exponentielle du nombre de services et des fournisseurs de services. Dans cette mouvance, la sélection des meilleurs services qu'ils soient atomiques ou composites représente un défi réel. L'apparition, disparition des services et la variation de leur qualité nécessitent des méthodes avancées pour la sélection dynamique des meilleurs services.

La problématique de notre travail concerne la sélection globale de services basée sur la qualité de service. Cette sélection, nécessite la prise en compte de plusieurs facteurs, qui sont: les exigences et les préférences locales et globales de l'utilisateur et le dynamisme des services, qui nécessitent une sélection à l'exécution et la prise en compte des informations du contexte. La prise en compte de ces facteurs pour la sélection de services est considérée comme un problème NP-complet.

Dans ce travail de recherche, nous proposons une approche basée sur le formalisme CSP pour la sélection du meilleur service tout en vérifiant les contraintes locales et globales.

Ce mémoire est organisé en quatre chapitres. Le premier chapitre qui constitue une introduction aux systèmes ubiquitaires, présente les défis de ce nouveau domaine en l'illustrant à travers des scénarii. Dans le deuxième chapitre, nous donnons tout d'abord un ensemble de définitions relatives aux fonctions essentielles d'un système ubiquitaire puis nous nous focalisons sur la problématique de la sélection de services qui est au cœur de notre travail. Nous passons également en revue les principales approches de sélection décrites dans la littérature. Le troisième chapitre détaille notre approche de sélection globale, qui se base sur le formalisme CSP. Le quatrième chapitre décrit la conception, l'implémentation et la mise en ouvre de notre approche. Dans ce chapitre, les performances de l'approche proposée sont également présentées et analysées.

Enfin, nous concluons ce mémoire par une conclusion générale et quelques perspectives.

1

Introduction aux systèmes ubiquitaires

I.1. Introduction

Grâce aux progrès réalisés dans la miniaturisation de composants électroniques et à l'essor des technologies de communication sans fil, un nombre croissant d'objets de notre quotidien intègre des dispositifs électroniques à l'aide desquels ils deviennent communicants **[Want, 2005]**. Ces *objets communicants* peuvent être des dispositifs informatiques, des périphériques, ou des équipements de domaines aussi variés que l'électroménager, les télécommunications, la domotique ou l'automobile **[Saha, 2003]**. Ces objets sont mobiles ou immobiles. Ils sont *mobiles* lorsqu'ils sont portés (téléphones, PDA, ordinateurs portables, etc.) ou conduits (dispositifs embarqués dans un véhicule) par leur utilisateur. Ils sont, en revanche, *immobiles* et invisibles lorsqu'ils sont intégrés ou enfouis dans des objets fixes de l'environnement de l'utilisateur (télévisions, lampes, machines à laver, réfrigérateurs, etc.).

Tout objet physique de l'environnement de l'utilisateur peut être potentiellement communicant et être ainsi capable d'interagir avec l'utilisateur ou avec d'autres objets. Chacun de ces objets dispose de sa propre unité de traitement, puissance de calcul, capacité mémoire, et connectivité réseau. L'utilisateur se retrouve finalement au centre d'un espace composé d'objets physiques hétérogènes, dotés de capacités de communication filaire ou non filaire. Ces objets doivent tous pouvoir communiquer avec l'utilisateur ou avec d'autres objets aussi bien localement qu'à distance. L'hétérogénéité des objets ne doit pas être un handicap à

leur collaboration qui doit être dynamique et spontanée dans le sens où : (i) elle ne doit pas être prévue à l'avance et (ii) elle doit s'adapter aux changements pouvant survenir dans l'environnement de l'utilisateur par l'apparition ou la disparition d'objets.

L'objectif est de permettre à l'utilisateur d'accéder aux différentes fonctionnalités offertes par les divers objets présents dans son environnement immédiat, à partir de n'importe quel terminal (PC portable, téléphone mobile, PDA : *Personal Digital Assistant*, etc). Les fonctionnalités offertes par ces objets peuvent être abstraites sous forme de services **[Papazoglou, 2003].** L'informatique diffuse consiste ainsi à offrir aux utilisateurs un accès aux services numériques de leur environnement immédiat quelle que soit l'hétérogénéité matérielle et logicielle des objets ou des dispositifs informatiques sous-jacents. Les architectures logicielles orientées services sont conçues pour supporter directement la vision de l'informatique ubiquitaire. Elles fournissent les bases permettant de réaliser des systèmes ubiquitaires par découverte, sélection et composition de services. Le présent chapitre constitue une introduction aux systèmes ubiquitaires de façon générale.

I.2. Définition d'un environnement ubiquitaire

L'informatique diffuse peut être vue comme un domaine particulier de l'informatique résultant de la convergence des travaux existants dans les domaines de l'informatique mobile et des systèmes distribués. La figure I.1 donne un aperçu des apports des systèmes distribués, de l'informatique mobile ainsi que les avancées nécessaires à la réalisation d'un système de l'informatique diffuse.

Les systèmes distribués constituent la rencontre entre les ordinateurs personnels et les réseaux locaux. Ils permettent le partage des capacités et des ressources à travers un réseau et une infrastructure de communication. Cela constitue la première étape de l'informatique diffuse par l'introduction de l'omniprésence de l'information.

L'informatique mobile est le résultat de l'intégration de la technologie cellulaire et du web pour donner la possibilité aux utilisateurs d'accéder à l'information à n'importe quel endroit et en utilisant différents équipements de petites tailles et de faibles coûts. C'est en quelque sorte la deuxième étape vers l'informatique diffuse caractérisé par le paradigme « n'importe où, n'importe quand » **[Satyanarayanan, 2001].**

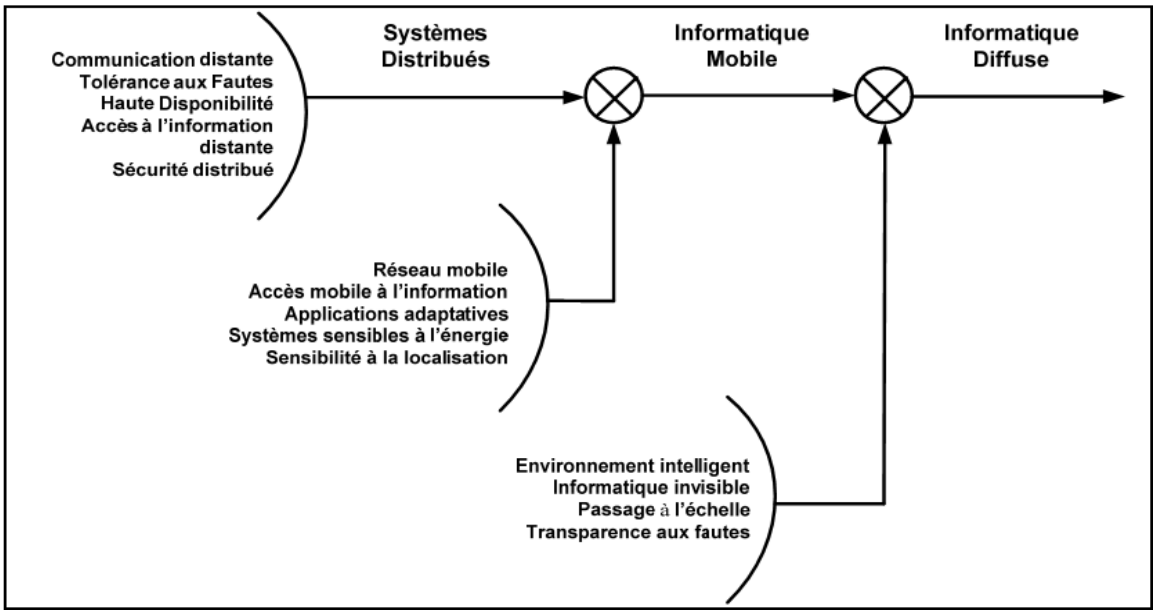


Figure I.1. Évolution vers l'informatique diffuse (adaptée de **[Satyanarayanan, 2001]**).

Contrairement à l'informatique traditionnelle qui suppose qu'un utilisateur effectue une tâche définie dans un environnement déterminé, l'informatique diffuse repose sur la connaissance du contexte de l'usager pour lui fournir le service approprié au moment opportun. Ainsi, le badge actif (ActiveBadge) **[Want, 1992b]**, inaugure l'une des premières applications de l'informatique diffuse. Cette application permet l'accès par identification à certains bureaux du PARC (Palo Alto Research Center / Centre de recherches de Xerox) et localise un usager afin de lui transférer ses appels téléphoniques vers le bureau où il se trouve. Depuis, les applications basées sur la géolocalisation se sont multipliées, et le guidage routier par GPS (Global Positioning System) en est certainement l'illustration la plus connue.

Le contexte ne peut cependant se réduire à la seule localisation d'un usager dans l'espace. D'une manière plus générale, il doit être envisagé **[Dey, 2000]** comme *« l'ensemble des informations qui peuvent être utilisées pour caractériser la situation courante d'une entité, telle qu'un individu, un lieu ou un objet considéré comme ayant un rapport avec le service fourni par le système »*.

Le projet *MyCampus* **[Sadeh, 2002]** exploite ainsi différentes données issues de l'environnement pour définir le contexte courant de l'usager. Ce système en vigueur sur le campus du CMU (Carnegie Mellon University) est composé de plusieurs « agents » qui remplissent des fonctions spécifiques. L'agent « concierge » indique dans quel restaurant universitaire l'étudiant peut prendre ses repas en fonction de ses préférences gastronomiques,

de l'heure, de sa localisation sur le campus et des conditions météorologiques. L'agent « réunion » illustre quant à lui, un autre aspect de l'informatique ambiante : celui de l'échange direct de machines à machines ; il assiste les usagers dans le choix d'une date commune de réunion, en parcourant leurs agendas respectifs. Plus généralement, les applications ambiantes relèvent de l'assistance discrète et permanente d'un usager dans l'ensemble de ses activités courantes **[Pascoe, 1998]**. En ce sens, elles dépassent le cadre actuel de l'informatique en mobilité.

Pour éviter tout risque de confusion entre les termes utilisés dans le domaine de l'informatique diffuse, nous donnons les définitions suivantes qui le caractérisent **[Saha, 2003]** :

- Ubiquitaire : Accessible de n' importe où ;
- Mobile : Qui intègre les terminaux mobiles ;
- Sensible au contexte : Qui prend en compte le contexte d'exécution ;
- Pervasif : Qui associe ubiquité, mobilité et sensibilité au contexte ;
- Ambiante : Qui est intégré dans les objets quotidiens.

I.3. Scénarii illustrant l'informatique ubiquitaire

Pour rendre clairs les concepts fondamentaux des systèmes ubiquitaires (diffus), nous présentons quelques exemples de scénarii concrêts dans lesquels un utilisateur interagit avec différents services de son environnement. Ensuite, nous procédons à une analyse de ces scénarii pour expliquer et mettre en évidence le fonctionnement de ces systèmes ainsi que leurs composants.

I. 3.1. Scénario 1

Dans ce scénario, nous supposons qu'un système ubiquitaire est installé dans le pavillon A d'une école **[Miraoui, 2009]**. *En entrant dans ce pavillon pour assister à un cours qui démarrera dans dix minutes et qui durera trois heures, Paul reçoit un appel sur son téléphone cellulaire de la part de son directeur de thèse pour lui rappeler qu'aujourd'hui (et avant deux heures) est la date limite pour envoyer trois rapports (documents volumineux sauvegardés dans son ordinateur portable) à son codirecteur de thèse.*

Tout calcul fait, Paul se rend compte qu'il est dans l'obligation de faire cette tâche avant d'assister au cours et sans s'absenter. Il se dirige alors vers la cafétéria et s'authentifie pour l'accès au réseau sans fil en utilisant son ordinateur portable. Il ouvre sa boîte de courriel et rédige le message qui accompagne les rapports. Lors de la jointure de son premier document, un message apparaît sur son écran lui signalant que d'après le débit actuel du réseau, l'expédition sera trop lente parce que plusieurs étudiants à la cafétéria naviguent sur internet. Le système remarque que le débit à la bibliothèque est très bon et qu'il y a peu d'étudiants qui utilisent internet, alors il propose à Paul de se déplacer vers la bibliothèque se trouvant à deux minutes de la cafétéria. Paul accepte la proposition et se dirige vers la bibliothèque.

En entrant dans celle-ci, son téléphone cellulaire se met automatiquement en mode vibreur pour respecter les règles du lieu. Paul procède rapidement à l'attache des trois documents à son message. Au cours de cette opération, il reçoit une alerte SMS qui s'affiche automatiquement cette fois sur son téléphone cellulaire indiquant qu'il lui reste trois minutes avant le prochain cours. Paul termine sa tâche et quitte le lieu en route en direction de la salle de cours.

Analyse du scénario

D'après le scénario, on peut facilement dégager les caractéristiques fondamentales d'un système informatique diffus. Il offre :

- Un service transparent sur un réseau, en particulier sur internet pour différentes tâches (naviguer ou expédier un courriel) ;
- Des équipements connectés par différents moyens (WiFi, Bluetooth, etc.) ;
- Un système qui surveille le contexte et les paramètres du service qu'il offre aux utilisateurs.

Le système doit s'adapter aux différents contextes pour aider l'utilisateur dans ses tâches et lui permettre de se concentrer sur ces tâches principales sans se préoccuper des moyens pour les accomplir.

1. Pour offrir un meilleur service à Paul, le système détecte que le débit de la connexion internet est trop faible et propose une solution. La qualité de la communication ici est

un élément important pour le choix du service, ainsi que l'activité (expédition d'un courriel contenant des documents volumineux) ;

2. Le type de service dépend dans ce scénario de la localisation de l'utilisateur (signalisation des appels pour le téléphone cellulaire avec mode sonnerie ou mode vibreur) ainsi que le service lui-même (alerte par bip sonore ou par SMS).

3. La communication entre les équipements est aussi présente dans notre scénario. À l'entrée dans la bibliothèque, le téléphone cellulaire est mis automatiquement en mode vibreur (le téléphone cellulaire est équipé d'un système GPS et est programmé pour changer de mode à l'intérieur de la bibliothèque où le système de surveillance de la bibliothèque envoie des requêtes aux cellulaires à l'entrée de la bibliothèque pour changer de mode et ces derniers sont programmés pour accepter ces requêtes).

4. De même, la communication entre le téléphone cellulaire et l'ordinateur portable (le téléphone cellulaire en mode vibreur à l'intérieur de la bibliothèque est susceptible d'alerter Paul du temps restant pour le prochain cours. Il communique alors avec son ordinateur portable pour voir s'il est en train de l'utiliser. Si c'est le cas, il envoie une alerte SMS).

5. La stratégie générale de découverte d'activité utilisée par les systèmes informatiques diffus consiste en la lecture d'une base de données (le cas de l'alerte du téléphone cellulaire pour indiquer le temps restant avant le prochain cours. L'emploi du temps de Paul est inscrit dans l'agenda interne du téléphone cellulaire qui constitue une source d'informations).

I.3.2. Scénario 2

Dans ce scénario, un utilisateur appelé Paul interagit avec différents services de son environnement **[Yérom-David, 2006]**. *En effet, Paul participe à une réunion qu'il doit malheureusement 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 routier, auquel le véhicule a accès via une infrastructure réseau, 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 le trajet.

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 Wi-Fi 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, faute de réseau sans fil, au réseau local, afin de découvrir un service permettant de projeter sur un écran la présentation de son projet. Il termine son rendez-vous en synchronisant des documents de son portable avec ceux d'un dispositif de stockage de données appartenant à Suzanne.

Analyse du scénario

Ce scénario met en valeur plusieurs concepts clés de l'informatique ubiquitaire, dont celui de permettre aux utilisateurs d'accéder à des ressources et services numériques n'importe où et à tout instant, à travers différents dispositifs. 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. Un autre principe clé de l'informatique ubiquitaire est l'aspect dynamique de l'environnement de l'utilisateur. Par exemple, l'accident annoncé est un évènement que l'on ne peut pas prédire. Cela a amené Paul à utiliser son PDA pour analyser son environnement immédiat afin, éventuellement, de trouver un service lui permettant d'obtenir un plan du quartier où il se trouve.

Un point important à souligner est que le développement de ces différents concepts de l'informatique ubiquitaire n'est possible qu'à la seule et unique condition que les différents dispositifs, terminaux ou services de l'environnement de l'utilisateur soient capables d'interagir indépendamment de leurs hétérogénéités matérielles et logicielles. Dans notre exemple, le PDA, le téléphone mobile, les dispositifs informatiques embarqués, et l'ordinateur portable sont soumis potentiellement à des contraintes matérielles différentes et

sont supportés par des systèmes logiciels hétérogènes. Ainsi, un des défis de l'informatique diffuse est donc d'intégrer, de façon transparente pour l'utilisateur, des technologies hétérogènes **[Satyanarayanan, 2001]**, **[Saha, 2003].**

I.4. Les défis de l'informatique ubiquitaire

Nous analysons dans ce paragraphe les problématiques les plus importantes qui caractérisent l'informatique ubiquitaire.

I.4.1. L'hétérogénéité

Les équipements utilisés dans un environnement ubiquitaire sont très variés (ordinateur portable, PDA, appareil électroménager, téléphone intelligent ou Smartphone, appareil médical, etc.). Ces dispositifs hébergent une variété de services, fonctionnent à base de différents systèmes d'exploitation et possèdent, généralement, des configurations matérielles et logicielles différentes. De plus ces équipements s'appuient sur diverses technologies de communication filaires ou no filaires (ADSL, GSM, GPRS, UMTS, Wifi, Bluetooth, etc.). Par exemple, un assistant numérique personnel (PDA pour Personal Digital Assistant) héberge un service de notifications d'alertes qui peut interagir avec d'autres services via une connexion Wifi ou Bluetooth.

I.4.2. Limitation des ressources

La miniaturisation est l'un des facteurs qui a parmi le développement de l'informatique mobile. La plupart des équipements mobiles sont conçus de manière à être facilement transportables par les utilisateurs. Ils sont petits en termes de forme, puissance de traitement, capacité de stockage, taille d'affichage et autonomie de batteries. Tous ses facteurs limitent le type de traitements effectués sur ces équipements. Malgré les progrès technologiques, les capacités de ces dispositifs resteront, sans doute, toujours limitées par rapport aux PC de bureau.

I.4.3. Dynamicité

Les systèmes ubiquitaires sont caractérisés par un environnement physique dont l'évolution est imprévisible. Notamment, la mobilité des utilisateurs et la volatilité des objets communicants rendent ces systèmes difficiles à appréhender. Il existe deux formes de

dynamicité dont les systèmes ubiquitaires sont dépendants : la dynamicité des objets communicants et celle du contexte.

a) Objets communicants

Sans être attachés à des équipements fixes, dans un environnement ubiquitaire, les utilisateurs peuvent se déplacer librement d'une place à une autre. Par exemple, un utilisateur dans sa voiture, utilise un afficheur numérique pour regarder une conférence. En quittant sa voiture vers la maison, l'utilisateur transfère la conférence à son PDA. Une fois arrivé à la maison, il continue à regarder la vidéo conférence sur son PC. Cette mobilité provoque un changement continu (dynamicité) de l'environnement et peut affecter l'exécution de certaines applications.

b) Contexte

Les systèmes ubiquitaires doivent être sensibles à leur environnement afin de prendre les décisions appropriées et fournir les services adéquats aux utilisateurs. Cette sensibilité consiste en la prise en compte de l'évolution des paramètres physiques de l'environnement. Ces paramètres constituent le contexte de l'environnement.

La mobilité dans un environnement ubiquitaire provoque des changements dans le contexte des applications. En effet, les premières applications sensibles au contexte en informatique ubiquitaire, étaient sensibles à la localisation. De façon générale, le contexte décrit la situation de l'utilisateur en termes de localisation, de temps, d'environnement, de terminal utilisé, de profil utilisateur, etc. Par exemple, une situation contextuelle peut être définie avec les paramètres suivants : profil utilisateur = *«infirmier»*, terminal utilisé = *«PDA»*, localisation = *«domicile du patient P »*. Le changement d'une valeur sur l'un de ces paramètres définit une nouvelle situation contextuelle à laquelle l'application doit s'adapter. On parle alors de dynamicité du contexte.

I.4.4. Criticité

En positionnant l'utilisateur au centre des préoccupations, les systèmes ubiquitaires doivent être robustes pour ne pas interférer négativement sur les activités quotidiennes des utilisateurs. De plus, l'informatique ubiquitaire touche des domaines applicatifs potentiellement critiques tels que le domaine hospitalier **[IBM, 2006]** et le domaine de la

gestion des accidents autoroutiers **[Costa, 2007].** Il est donc primordial de s'assurer de la robustesse des systèmes ubiquitaires en situation réelle et ceci quelque soit le scénario, c'est-à-dire, quelque soit l'évolution particulière du contexte et des services sur une période donnée. Par exemple, un hôpital ubiquitaire doit être capable de gérer un important flux de blessés suite à un accident de la route.

I.6. Conclusion

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

2

Sélection de services. Définitions et état de l'art

II.1. Introduction

Dans ce chapitre, nous donnons tout d'abord un ensemble de définitions relatives aux fonctions essentielles d'un système ubiquitaire puis nous nous focalisons sur la problématique de la sélection de services qui est au cœur de notre travail. Nous passant passons également en revue les principales approches de sélection présentées dans la littérature.

II.1.1. Définition d'un service

Un service est une entité logicielle qui permet d'exposer une ou plusieurs fonctionnalités définies par une interface **[Chollet, 2009]**. Cette interface comporte les propriétés fonctionnelles et éventuellement des propriétés non-fonctionnelles d'un service permettant d'identifier le service qui convient le mieux aux besoins des utilisateurs.

Un service est dit composite lorsque son exécution nécessite d'invoquer plusieurs autres services en faisant appel à leurs fonctionnalités, sinon il est dit atomique.

II.1.2. Les propriétés d'un service

L'identité d'un service est définie par ses propriétés fonctionnelles et non-fonctionnelles. En effet, les services se distinguent par les fonctionnalités qu'ils peuvent fournir. Deux services équivalents, ç.à.d. offrant sémantiquement les mêmes fonctionnalités (exemple un service « Moisture » et un service « Humidité » qui offrent des fonctions de mesure d'humidité), peuvent être différenciés par leurs propriétés non-fonctionnelles. Par

exemple le service « Humidité » a un temps d'exécution inférieur au service « Moisture » alors que le service « Moisture » est caractérisé par un taux de consommation d'énergie inferieur au premier. Les propriétés non-fonctionnelles sont la base de définition de la qualité de service.

II.1.2.1. Les propriétés fonctionnelles

Les propriétés fonctionnelles d'un service désignent les fonctionnalités que ce service peut fournir. Les propriétés fonctionnelles sont décrites dans la description de service en termes d'opérations, et reflètent le fonctionnement du service, comme par exemple un service de localisation.

II.1.2.2. La qualité de service

La Qualité de service (Quality of Service ou QoS) est l'aptitude d'un service à répondre adéquatement et dans de bonnes conditions aux exigences de l'utilisateur, dans le but de le satisfaire. Ces exigences peuvent être liées à plusieurs propriétés non-fonctionnelles d'un service à savoir: la disponibilité, la fiabilité, le temps de réponse, la sécurité, etc.

II.1.3. Définition de l'approche à services

L'approche à services (ou SOC pour *Service Oriented Computing*) est un paradigme informatique qui repose sur la construction d'applications à partir de services. Ces services sont fournis par des organisations tierces et peuvent évoluer dynamiquement **[Papazoglou, 2003]**. Cette approche repose sur trois acteurs (Fournisseur de services, Registre de services et Consommateur de services) et trois interactions (Publication, Découverte et Invocation), comme le montre la figure II.1.

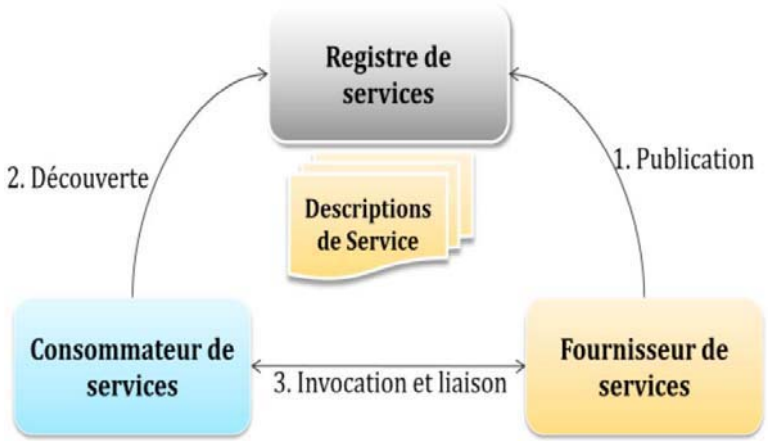


Figure II.1. Interaction entre les acteurs de l'approche à services.

II.1.4. Les acteurs de l'approche à services

Le tableau II.1, présente les trois acteurs de l'approche à services selon l'ordre chronologique de la mise en œuvre d'une application à services.

Acteur	Le rôle
Fournisseur de services.	Le fournisseur de service a pour rôle de fournir des services avec leurs descriptions. Ces descriptions sont publiées dans un registre afin de permettre la découverte de ces services.
Registre de services.	Le registre joue le rôle d'intermédiaire entre les fournisseurs et les consommateurs des services. Il stocke des descriptions de services qui référencent leurs fournisseurs.
Consommateur des services.	Le consommateur de service est le client qui demande un service. Pour ce faire, le consommateur doit être lié au fournisseur du service après sa découverte dans le registre.

Tableau II.1. Acteurs de l'approche à services.

II.1.5. Les interactions de l'approche à services

Les interactions de l'approche à services échangées entre les acteurs de l'approche sont décrites dans le tableau II.2.

Interaction	Définition
La publication.	Un fournisseur de service s'enregistre dans l'annuaire (registre) de service pour publier sa description de service.
La découverte.	La découverte est une primitive qui consiste à rechercher des services qui repondent aux besoins des utilisateurs.
L'invocation.	L'invocation est réalisée après la découverte des services. Cette primitive consiste à établir une liaison entre le consommateur et le fournisseur pour permettre son utilisation.

Tableau II.2. Interactions de l'approche à services.

II.1.6. Les caractéristiques de l'approche à services

L'approche à services est caractérisée par le principe de faible couplage induit par l'indépendance des services. L'objectif est d'introduire le minimum de dépendances entre les services, et pouvoir les assembler facilement. Ce faible couplage favorise la réutilisation des services et leur composition. Il permet aussi de masquer l'hétérogénéité en cachant les détails d'implémentation des services. Un autre point fort de cette approche est la liaison tardive entre le fournisseur et le consommateur. En effet, la liaison entre ces derniers est établie lorsque le service demandé par l'utilisateur est trouvé suite à la requête de ce dernier.

II.1.7. L'approche à services dynamiques

Nous avons présenté jusqu'ici l'approche à services sans tenir compte du changement de contexte **[Redmond, 2002]**. Ce changement de contexte peut se traduire par :

II.1.7.1. La mise à jour d'un service

La mise à jour concerne principalement les propriétés non-fonctionnelles qui peuvent être contenues dans la description d'un service (exemple : le temps de réponse d'un service donné). Un service peut annoncer dans sa description la valeur de son temps de réponse, et au fil des excusions, ce temps de réponse peut augmenter ou diminuer. Cette capacité de changement peut avoir des répercussions comme le changement d'un service par un autre ayant un temps de réponse meilleur et qui est disponible dynamiquement.

II.1.7.2. L'apparition d'un nouveau service

Les nouveaux services peuvent être publiés pendant l'exécution de l'application. Le consommateur du service est donc notifié, comme le montre la figure II.2.

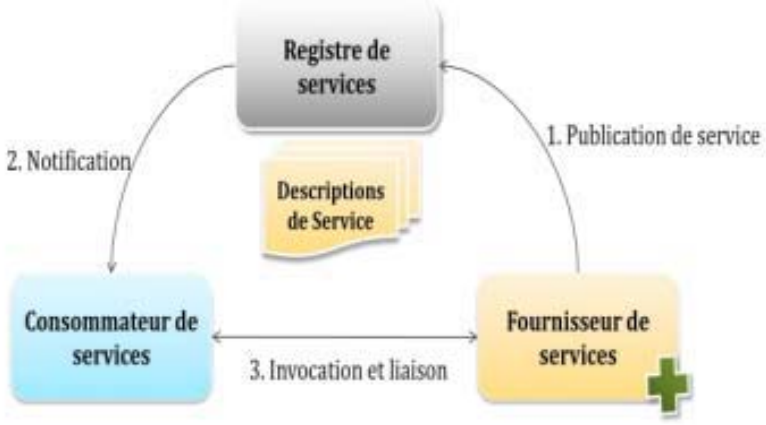


Figure II.2. L'apparition d'un nouveau service.

II.1.7.3. La disparition d'un service

Un service peut être retiré à tout moment du registre des services. Le consommateur est informé de cette disparition par une notification envoyée par le registre (figure II.3).

Figure II.3. Disparition d'un service.

II.2. La composition de services

II.2.1. Définition de la composition de services

La composition de services consiste à combiner les fonctionnalités de plusieurs services dans le but de répondre à des demandes complexes qu'un seul service ne pourrait pas satisfaire **[Mrissa, 2007]**. La composition de services vise à faire intéropérer, interagir et coordonner plusieurs services pour la réalisation d'un but donné **[Charif, 2007].**

La figure II.4 illustre le principe de la composition de services à partir d'un ensemble de services disponibles dans un registre (annuaire).

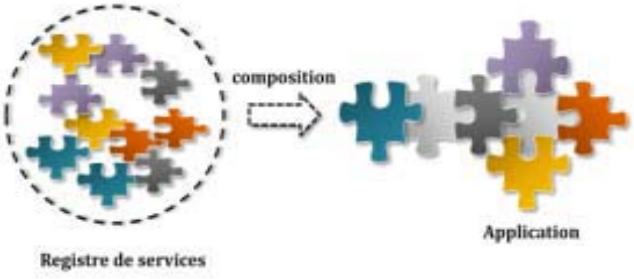


Figure II.4. Principe de composition de services.

II.2.2. La classification des méthodes de composition de services

La classification des méthodes de composition peut s'effectuer de deux façons : (i) en fonction de degré d'implication de l'utilisateur dans la définition du schéma de composition, et dans ce cas, les méthodes de composition peuvent être manuelles ou automatiques. (ii) Selon la sélection des services, soit faite a priori ou non. Dans ce cas, l'approche sera dite statique ou dynamique.

II.2.2.1. Composition statique / dynamique

La composition statique est caractérisée par le fait que les interactions entre les services sont spécifiées dès le départ contrairement à la composition dynamique des services dont l'objectif est de produire un plan d'exécution automatique en utilisant les descriptions des fonctionnalités des services existants.

II.2.2.2. Composition manuelle / automatique

La composition manuelle repose sur un expert dont la mission est de définir et générer des scripts des processus métier qui sont ensuite soumis à un moteur d'exécution. La composition dynamique, quant à elle, automatise entièrement le processus de composition en fonction de la requête de l'utilisateur.

II.3. La sélection de services

La sélection de service constitue une étape importante dans la composition de services. Le processus de composition commence par une requête de l'utilisateur sous forme d'un service abstrait composite. Le module de découverte utilise le registre de services (l'annuaire) afin de trouver les services disponibles pour chaque service abstrait dans le plan abstrait fourni par l'utilisateur, en effectuant une correspondance fonctionnelle syntaxique (et éventuellement sémantique) entre les services abstraits et les descriptions des services disponibles. Par conséquent, une liste de services concrets est obtenue pour chaque service abstrait. L'objectif de la sélection de services avec QoS est de choisir un service concret de chaque servie abstrait de telle sorte à avoir une QoS qui satisfait les exigences de l'utilisateur, comme le montre la figure II.5.

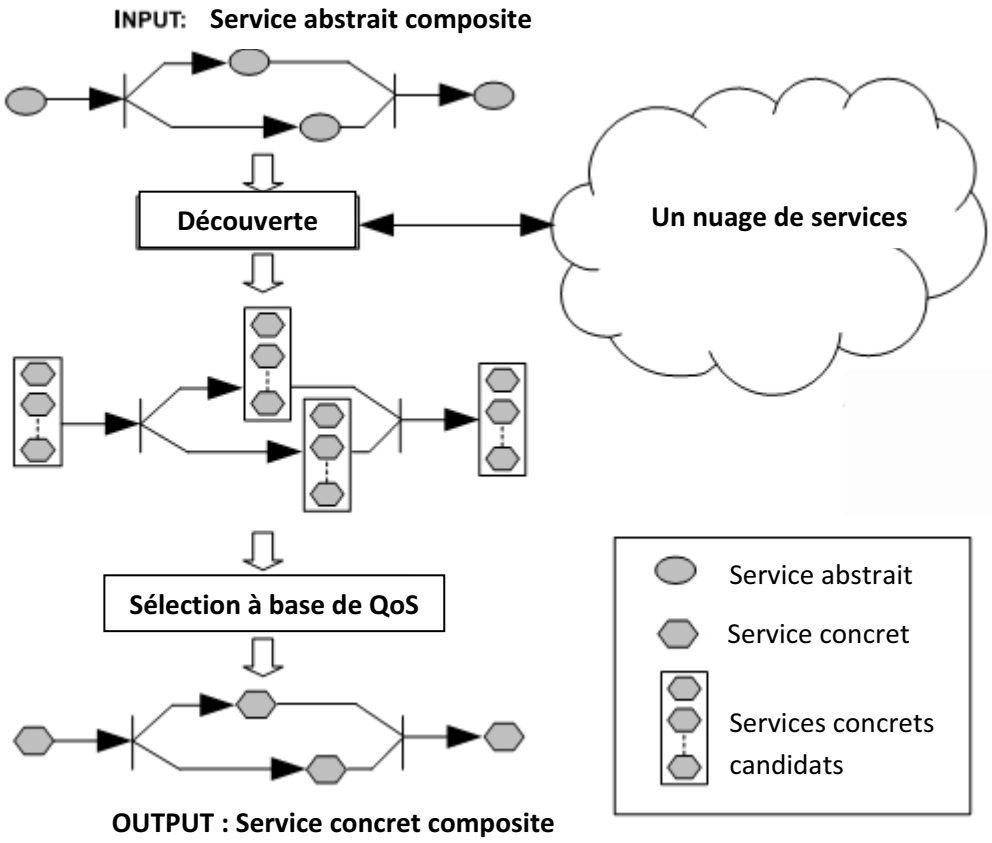


Figure II.5. Schéma global de sélection.

II.3.1. Service concret

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

CSi ^{in} : sont les paramètres d'entrées du service.

CSi ^{out} : sont les paramètres de sortie du service.

Prec : les pré-conditions du service.

Eff : les effets du service.

Cxt : l'ensemble des attributs du contexte auxquels le service est sensible.

QoS : les paramètres de qualité du service.

La figure II.6 montre une représentation d'un service concret.

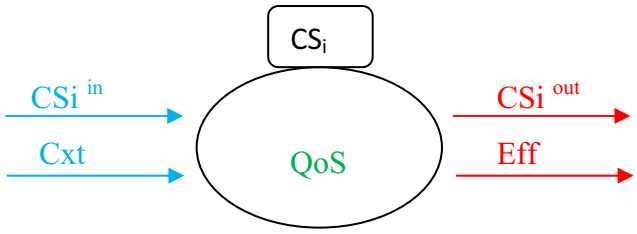


Figure II.6. Représentation d'un service concret.

II.3.2. Service abstrait

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

ASi in : sont les paramètres d'entrée du service abstrait ASi.

ASi out : sont les paramètres de sortie du service abstrait ASi.

CS : l'ensemble des services concrets.

La figure II.7 illustre la représentation d'un service abstrait.

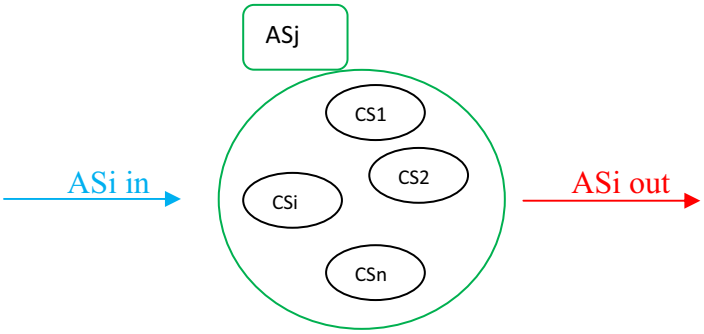


Figure II.7. Représentation d'un service abstrait.

II.3.3. Définition de la sélection des services

La sélection de services consiste à sélectionner le meilleur service répondant aux besoins de l'utilisateur. La sélection se base principalement sur la description des services qui regroupe un ensemble de fonctionnalités et un ensemble de paramètres de qualités pour qualifier chaque service.

II.3.4. La sélection de services selon les propriétés fonctionnelles

La sélection de services selon les propriétés fonctionnelles se base sur l'adéquation syntaxique exacte entre l'interface de service fournie par les fournisseurs et l'interface de service requise par le consommateur. Cette technique d'adéquation syntaxique s'avère parfois insuffisante au regard de la diversité des requêtes des utilisateurs et des technologies utilisées. D'où l'apparition du principe de sélection sémantique. Malgré cette évolution, la sélection basée seulement sur les propriétés fonctionnelles, présente une insuffisance. En effet, dans de nombreux cas, l'utilisateur se trouve dans des situations où il doit choisir un service parmi plusieurs services équivalents qui offrent sémantiquement les mêmes

fonctionnalités. Par conséquent, l'utilisateur doit savoir non seulement ce que fait le service (propriétés fonctionnelles) mais aussi comment il le fait (la qualité de service) en se basant sur les propriétés non-fonctionnelles du service.

II.3.5. La sélection des services basée sur la qualité de service

La sélection des services basée sur la qualité de service fait l'hypothèse que les services qui feront l'objet de la sélection répondent tous aux besoins fonctionnels de l'utilisateur. Ainsi ces approches de sélection s'intéressent uniquement aux propriétés non-fonctionnelles afin de garantir la sélection du meilleur service en termes de qualité de service sur les plans local et global.

II.3.5.1. Sélection Locale

Le principe des approches de sélection locale est de choisir un service de chaque service abstrait indépendamment des autres services abstraits. En utilisant une fonction d'utilité, les valeurs des différents paramètres de QoS sont transformées en une seule valeur d'utilité. Le service possédant la valeur d'utilité maximale est sélectionné. Cette approche est très efficace en termes de temps de calcul du fait que la complexité en temps de cette approche est en O (n), où n est le nombre de services concrets associés à chaque service abstrait. L'inconvénient majeur de la sélection locale est qu'elle n'est pas adaptée à la composition de services basée sur la QoS, avec des contraintes globales comme par exemple, le temps de réponse global, étant donné que ces contraintes globales ne peuvent être vérifiées localement.

II.3.5.2. Sélection globale

La sélection globale est basée sur les approches d'optimisation globale qui ont été récemment mises en avant comme solutions au problème de sélection globale de services avec QoS **[Zeng, 2004]**, **[Ardagna, 2007]**. Ce type d'approches vise à résoudre le problème au niveau du service composite, en évaluant toutes les combinaisons possibles de QoS des services concrets, afin de choisir la meilleure combinaison en fonction des QoS, tout en satisfaisant les contraintes globales. Le problème de la sélection globale est vue comme un problème multi critères et NP-difficile, d'où la difficulté de trouver une solution optimale en un temps raisonnable **[Maros, 2003]**.

II.3.5.3. Les approches de sélection basées sur la qualité de service

Les approches de sélection de services basées sur la QoS sont très nombreuses et peuvent être classées en trois grandes classes :

a. Les approches basées sur la description sémantique

Les approches de cette catégorie se basent principalement sur le Web sémantique. Les propriétés non fonctionnelles sont définies en utilisant des ontologies **[Wang, 2006]**, D'autres approches s'intéressent à l'annuaire de services et proposent une extension de l'annuaire qui est vu comme un modèle d'expression des propriétés non fonctionnelles **[Eyhab, 2007]**. Le principal inconvénient d'une telle approche réside dans la séparation des descriptions de services et des informations associées aux propriétés non-fonctionnelles. De plus, le modèle de propriétés non-fonctionnelles n'est pas extensible et se restreint à quelques propriétés génériques telles que la disponibilité, le temps de réponse et le coût.

b. Les approches basées sur des mécanismes de réputation et de vote

Dans ces approches, la sélection de services utilise l'historique des services et les avis des utilisateurs **[Wang, 2007]**, comme l'illustre la figure II.8.

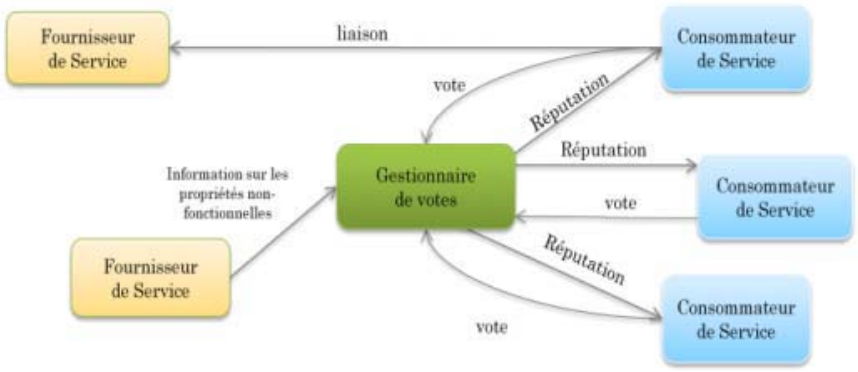


Figure II.8. Système de réputation et de vote.

L'inconvénient de cette approche est l'implication des utilisateurs pour assurer le système de vote, ce qui remet en cause la sélection automatique et nécessite une coopération basée sur la confiance entre les utilisateurs.

c. Les approches basées sur une modélisation graphique et analytique

Plusieurs travaux de recherche ont été menés dans ce cadre. Notre approche de sélection s'inscrit dans cette catégorie d'approches que nous détaillons ci-après.

Ces approches de sélection de service utilisent des travaux tirés de la recherche opérationnelle : CSP (problème de satisfaction de contraintes), théorie des graphes, programmation linéaire, etc.

Florian Rosenberg et al. proposent dans **[Rosenberg, 2009]** une formulation du problème de sélection sous forme d'un problème de satisfaction de contraintes (CSP), où les entrées sont l'ensemble des services concrets et les contraintes considérées sont les contraintes locales et globales. Le CSP peut avoir plusieurs solutions, et par conséquent, un poids est associé à chaque niveau hiérarchique différent de contraintes pour trouver la solution optimale.

Zhang et al. proposent dans **[Zhang, 2007]** deux modèles pour le problème de sélection de services composites : Le premier est un modèle combinatoire et le deuxième est un modèle de graphe. Ils ont utilisé une heuristique pour chaque modèle proposé. La complexité en temps de l'heuristique pour le modèle combinatoire est polynomiale, tandis que la complexité de l'heuristique pour le modèle de graphe est exponentielle. Bien que l'utilisation des heuristiques proposées permette de trouver une solution proche de l'optimum et soit plus efficace que des solutions exactes, les deux algorithmes ne sont pas adaptés pour un passage à l'échelle lorsque le nombre de services devient très élevé.

Mabrouk et al. proposent dans **[B. Mabrouk, 2009]** une approche heuristique. Ils utilisent les arbres de recherche pour la représentation et la sélection des services. Cette approche est basée sur la théorie d'utilité, qui utilise une méthode de clustering pour regrouper les services concrets dans k clusters. Cette fonction d'utilité dépend des valeurs de paramètres de qualité d'un service donné, et du nombre de services dans le cluster où se trouve le service en question.

Michael et al. **[Michael, 2007]** proposent aussi une approche heuristique basée sur un algorithme génétique. Les auteurs utilisent un modèle de propriétés de paramètres de qualité défini dans **[Michael, 2004]**. Ce modèle a pour objectif de transformer les paramètres de qualité des services concrets en paramètres de qualité de service composite, et cela en utilisant un workflow.

Ben Mokhtar et al. proposent dans **[B. Mokhtar, 2007]** COCOA, une solution pour la composition des services avec support de la QoS. Cette solution fournit COCOA–SD, un module de sélection des services basé sur les dépendances de contrôle inhérentes à la

spécification de composition de services. Par exemple, un service qui fournit une fonctionnalité sémantiquement équivalente à l'une des fonctionnalités demandée de la tâche utilisateur, ne pourrait pas être utilisé pour la composition si elle dispose des dépendances de données ou de contrôle avec des fonctionnalités qui ne sont pas demandées dans la tâche de l'utilisateur. Cette sélection est effectuée en utilisant une expression régulière correspondant au langage généré par l'automate de la tâche. Notons par L le langage généré par l'expression régulière et par L1, L2,. . . , Ln les langages générés par les automates des services présélectionnés S1, S2,. . . , Sn, respectivement. CACAO-SD sélectionne tous les services tels que $L \cap Li \neq ø$. Ceci permet la sélection des services qui répondent aux dépendances de contrôle de la tâche de l'utilisateur. En outre, si une dépendance de données est spécifiée entre deux fonctionnalités de la tâche de l'utilisateur, seuls les services qui offrent à la fois ces fonctionnalités dans leur composition sont conservés par les services précédemment sélectionnés. En plus de vérifier les différentes dépendances, une vérification de la conformité aux contraintes de QoS est effectuée. Ceci est fait en utilisant les formules de QoS définies dans **[B. Mokhtar, 2007]** selon la structure de l'automate.

Wang et al. proposent dans **[Wang, 2006]** une approche basée sur un algorithme de routage. Cette approche modélise la composition de services par un graphe pondéré qui définit l'ordre de sélection des services et associe à chaque arc une valeur de paramètre de qualité. Les nœuds du graphe représentent les services concrets et les arcs représentent les relations entre les différents services. La sélection des services se base sur l'algorithme de Dijkstra pour déterminer le chemin le plus court entre les services concrets constituant le service composite.

Alrifai et al. abordent dans **[Alrifai, 2009]** le problème de sélection de services locale tout en satisfaisant les exigences globales de l'utilisateur. Les auteurs proposent une solution combinant l'optimisation globale avec des techniques de sélection locale. La solution proposée comprend deux étapes : la première consiste à réaliser une décomposition des contraintes de QoS globales en contraintes locales d'une façon optimale, en utilisant la programmation entière mixte (Mixed Integer Programming). La deuxième étape consiste, quant à elle, à utiliser la sélection locale pour trouver les meilleurs services répondant aux contraintes locales. Dans ce travail, les attributs de QoS sont divisés en deux classes : attributs positifs et attributs négatifs. L'approche proposée ne considère que les attributs négatifs. Bien que la Qos d'un service composite soit décrite par la qualité des services qui le composent et

le modèle de composition utilisé (séquentiel, parallèle, conditionnel, et/ou boucle), les auteurs n'utilisent que le modèle séquentiel.

II.3.5.4. Synthèse

Nous présentons dans ce paragraphe une synthèse des approches de sélection analysées précédemment.

Approches	Travaux (références)	Principe utilisé	Caractéristiques
Approches basées sur la description sémantique	[Wang, 2007]	Web service sémantique et les ontologies.	- Ontologie complémentaire qui fournit plus de détails sur les aspects non fonctionnels. - Modèle extensible de propriétés non fonctionnelles.
	[Eyhab, 2007]	Extension de l'annuaire.	- Séparation des descriptions de services et des informations associées aux propriétés non fonctionnelles. - Modèle de propriétés non fonctionnelles non extensible.
Approches basées sur des mécanismes de réputation et de vote.	[Wang, 2007]	Gestionnaire de votes	- Observation et historique d'un service. -Coopération des utilisateurs. - Nécessité de confiance entre les utilisateurs.
	[Rosenberg, 2009]	Problème de satisfaction de contraintes (CSP).	- Prise en compte des contraintes locales et globales. -un poids est associé à chaque niveau hiérarchique différent de contraintes.

Approches basées sur une modélisation graphique et analytique	[Zhang, 2007]	-Utilisation d'heuristiques. - Modèle combinatoire. -Modèle de graphe.	-Sélection des services composites. -Complexité polynomiale de l'heuristique pour le modèle combinatoire. -Complexité exponentielle de l'heuristique pour le modèle de graphe. - solution proche de l'optimum. -Problème de passage à l'échelle.
	[B. Mabrouk, 2009]	- Arbre de Recherche - Méthode de clustering.	- Heuristique - Modèle de propriétés non fonctionnelles générique. - Evaluation de l'utilité d'un service par un algorithme de clustering.
	[Michael, 2007] et [Michael, 2004]	-Algorithme génétique.	- Heuristique - Modèle d'agrégation de paramètres de qualité.
	[B. Mokhtar, 2007]	-Automate à états finis. - Expression régulière.	- Dépendances de contrôle. - Sélection globale avec comme hypothèse la sélection locale. -Trois modèles de composition : séquentiel, boucle et choix.
	[Wang, 2006]	- Algorithme de routage pour la représentation des services. - Algorithme de Dijkstra pour la sélection.	Sélection d'un service atomique influencée par la sélection du service composite.
	[Alrifai, 2009]	-Programmation entière mixte pour la décomposition des contraintes de QoS globales en contraintes locales.	-Seuls les attributs négatifs sont considérés. - Seul le modèle séquentiel est utilisé.

		-Combinaison de l'optimisation globale avec des techniques de sélection locale.	

Tableau II.3. Synthèse des approches de sélection de services.

3

Approche de sélection globale de service, basée sur la QoS

III.1. Introduction

Ce travail se focalise sur le problème de sélection de services en fonction des préférences des utilisateurs et leurs exigences en termes de qualité de service (QoS).
Cette approche 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 l'annuaire des services concrets.
- Les services concrets sont classés en fonction de leurs tâches dans les services abstraits.
- Une mise à jour régulière est effectuée en fonction de la disparition ou de l'apparition des services concrets.

Afin d'apporter une solution à ce problème de sélection, nous avons proposé une approche intégrant plusieurs modules à savoir : 1) un module de spécification des exigences et des préférences de l'utilisateur, 2) un module d'évaluation de la qualité de service, 3) un module de sélection de service composite, 4) un module d'invocation de services et, 5) un

module de mise à jour des paramètres de qualité. La figure III.1 montre l'organisation de ces différents modules.

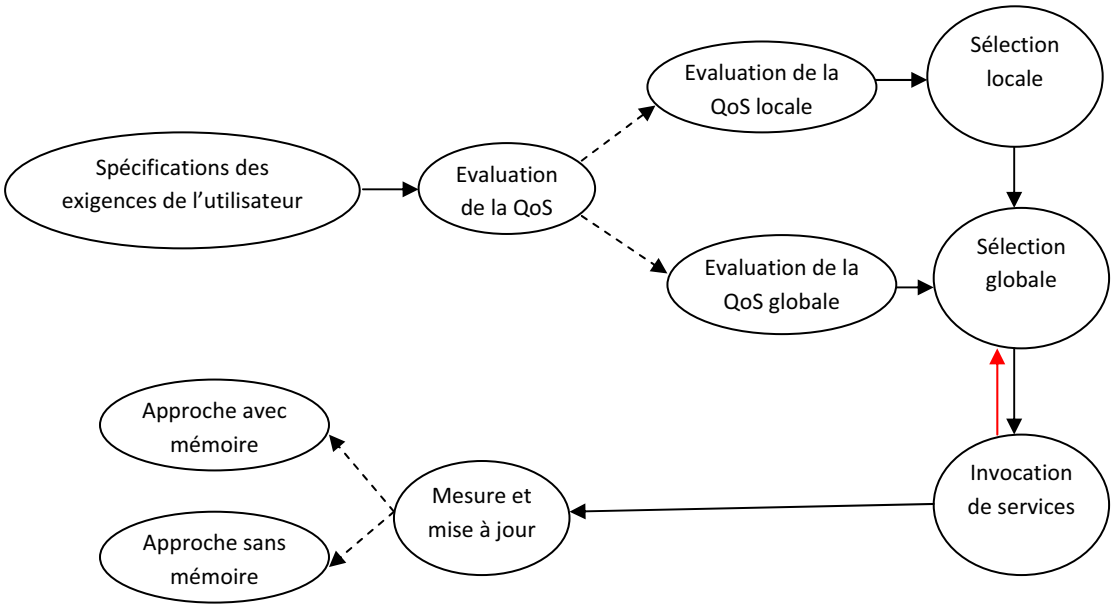


Figure III.1. Organisation des modules de l'approche de sélection de services.

La première étape dans le processus de sélection de services consiste à spécifier toutes les préférences et exigences des utilisateurs. Les préférences reflètent l'importance qu'accorde l'utilisateur à certains paramètres de qualité de service par rapport à d'autres, tandis que les exigences posent des contraintes sur certains paramètres de qualité. Ainsi, la sélection de services vise la satisfaction des préférences de l'utilisateur tout en prenant en compte les contraintes liées aux exigences spécifiées par ce dernier. Une fois les spécifications établies, la deuxième étape consiste à évaluer concrètement les paramètres de qualité de service. Cette évaluation est dite locale lorsqu'elle porte sur la QoS de chaque service atomique à l'intérieur de la même classe de services qui sont fonctionnellement équivalents. L'évaluation est dite globale lorsqu'elle porte sur la QoS d'un service composite. La troisième étape porte sur la sélection de services elle-même. Deux types de sélection sont alors considérés : la sélection locale et la sélection globale. La sélection locale choisi le meilleur service atomique dans une même classe de services équivalents, tandis que la sélection globale choisi plusieurs services atomiques pour un service composite en tenant compte de la structure de ce dernier. Il est évident que la sélection globale utilise la sélection locale au moins pour établir une classification à l'intérieur de chaque classe de services. La quatrième et dernière étape concerne l'invocation des services sélectionnés suivie d'une mise à jour (approche avec et sans mémoire) des différents paramètres de qualité. En cas d'échec lors de l'invocation d'un

service, l'étape de sélection peut être relancée de nouveau afin de trouver un service assurant le remplacement du service défaillant.

III.2. Spécifications de paramètres de qualité de service

Notre approche distingue deux types de paramètres de qualité de services, les paramètres statiques (***SQP*** pour **S**tatic **Q**uality **P**arameters) et les paramètres dynamiques (***DQP*** pour **D**ynamic **Q**uality **P**arameters).

III.2.1. Les paramètres de qualité statiques (SQP)

Les paramètres de qualité statiques (*SQP*) concernent les paramètres de qualité invariables durant une longue période de temps, ce que signifie que la fréquence de mise à jour des valeurs de ces paramètres est très petite. Ces paramètres sont généralement spécifiés lors de la publication d'un service. Dans ce qui suit, nous tenons compte de deux types de paramètres statiques à savoir: le coût et la sécurité.

a) Coût : ce paramètre peut être considéré suivant différents angles de vues. Il peut être vu, par exemple, comme le coût en termes d'énergie consommée, ou bien en termes d'espace mémoire nécessaire pour l'exécution du service, etc. Dans notre approche, ce paramètre est considèré comme étant le prix d'une prestation de service.

b) Sécurité : ce paramètre indique le niveau de sécurité assuré par le service. Un niveau de sécurité peut concerner le niveau de gestion de l'authentification et des autorisations, le niveau de chiffrement des données échangées, etc.

III.2.2. Les paramètres de qualité dynamiques (DQP)

Contrairement aux paramètres de qualité statiques, les paramètres dynamiques changent en fonction du contexte d'utilisation. Par exemple, la disponibilité d'un équipement peut dépendre de son niveau énergétique courant. Dans ce qui suit, nous considérons quatre paramètres de qualité dynamique : *la disponibilité (AV), le niveau d'énergie (EL), la fiabilité (RE)*, et le *temps de réponse (RT)*. (1) La *disponibilité* représente l'accessibilité d'un service lors de son invocation. (2) la *fiabilité (RE)* quantifie le nombre de paramètres reçus d'un service par rapport à l'ensemble de paramètres exigés, en tenant compte de l'état de chaque paramètre reçu. (3) le *temps de réponse (RT)* représente le temps requis par un service pour répondre à une demande. (4) le *niveau d'énergie (EL)* indique la quantité d'énergie à

consommer pour une prestation de service par rapport au niveau actuel de la batterie du dispositif hébergeant ce service.

III.3. Spécification des exigences de l'utilisateur

Afin de mieux répondre aux besoins de l'utilisateur par le module de sélection de services, il est nécessaire de spécifier les exigences de manière claire et compréhensible. Pour ce faire, nous distinguons quatre paramètres principaux de l'utilisateur (Figure III.2).

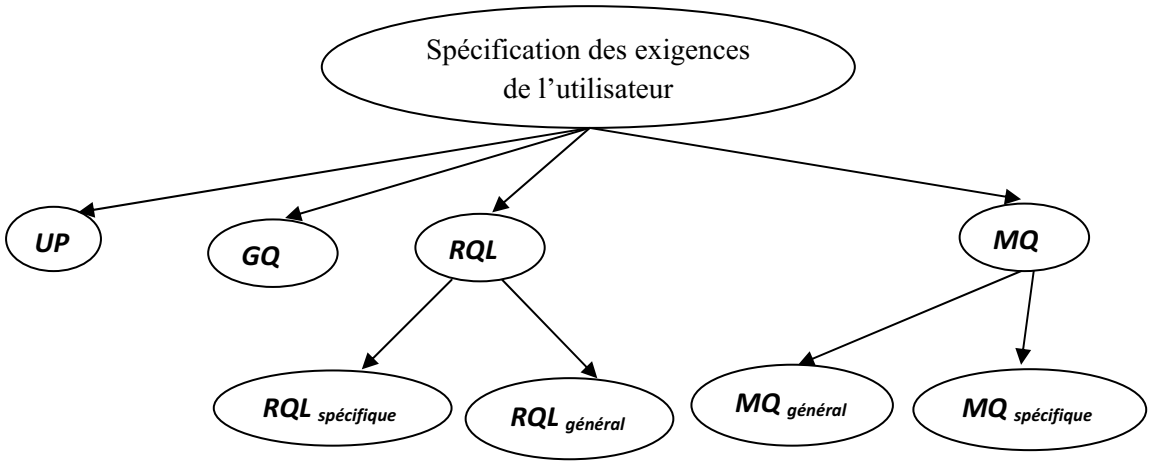


Figure III.2. Spécification des exigences de l'utilisateur.

III.3.1. La qualité obligatoire (MQ pour *Mandatory Quality*)

MQ comprend tous les paramètres de qualité devant **obligatoirement** être satisfaits. En d'autres termes, tout service n'offrant pas ces paramètres avec les valeurs exigées, sera rejeté de la liste des services à sélectionner. La qualité obligatoire inclut deux types de qualité : généraux et spécifiques.

a) ***Les paramètres généraux MQ*** $_{général}$

Ces paramètres doivent être satisfaits par chaque service ; la spécification de ce type de paramètres est effectuée pour l'ensemble de tous les services.

Par exemple, si on veut spécifier que chaque service doit avoir un niveau de sécurité supérieur à 3 (Sécurité (service $_i$) > 3, $\forall i$), il suffit donc d'exprimer cette exigences comme étant un paramètre générale ***MQ*** $_{général}$.

b) ***Les paramètres spécifiques MQ*** $_{Spécifique}$

Ils ne doivent être satisfaits que par le service spécifié (**AS** $_{spécifié}$). Par exemple, la spécification d'un niveau de sécurité supérieur à 3 pour le service AS (Sécurité (service $_{as}$) > 3).

Ainsi, chaque service AS doit satisfaire ***MQ*** $_{spécifique}$ quand elle est définie, sinon il doit satisfaire ***MQ*** $_{général}$.

III.3.2. Le niveau de qualité requis (RQL pour *Required Quality Level*)

RQL représente le niveau de qualité minimum requis ; sa valeur est comprise entre 0 et 1. ***RQL*** peut être défini de manière ***spécifique*** pour un service AS donné ou de manière ***générale*** pour tous les services. Ainsi, un service AS doit satisfaire son ***RQL Spécifique*** quand il est défini, sinon il doit satisfaire le ***RQL Général***.

III.3.3. La qualité globale (GQ pour *Global Quality*)

L'utilisateur spécifie ses exigences globales portant sur les valeurs d'un ensemble de paramètres de qualité, qui doivent être vérifiées par un service composite. Par exemple, EL $_{globale}$ < 60% et sécurité $_{globale}$ > 2, signifient que le niveau d'énergie global doit être inferieur à 60% et que le niveau global de sécurité doit être supérieur à 2.

Notre approche offre la possibilité à l'utilisateur de spécifier ses exigences par des valeurs exactes ou bornées (figure III.3). Par exemple **EL** $_{globale}$ < 40% et **sécurité** $_{globale}$ =3.

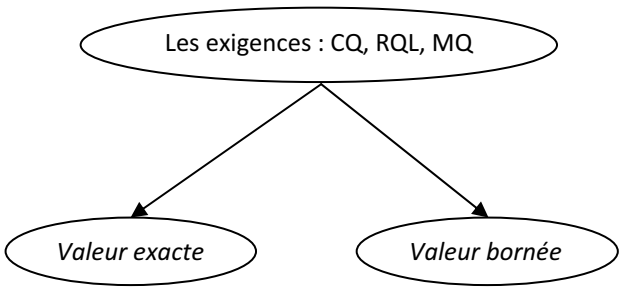


Figure III.3. Spécification, valeur exacte ou bornée.

III.3.4. Préférences de l'utilisateur (UP pour *User's Preferences*)

L'utilisateur spécifie ses préférences pour chaque paramètre de QoS en associant un poids wi à chaque paramètre ***i*** de QoS afin de refléter son importance relative.

Wi est compris entre 1 et 6 selon le degré d'importance (du moins important au plus important). Afin de faciliter à l'utilisateur la spécification de ses préférences, notamment les poids Wi, nous avons défini les correspondances suivantes :

Wi = 1 => très faible.

Wi = 2 => faible.

Wi = 3 => moyen.

Wi = 4 => assez élevé.

Wi = 5 => élevé.

Wi = 6 => très élevé.

Dans le cas où aucune spécification n'est fournie par l'utilisateur, nous considérons les paramètres par défaut comme suit : $MQ = \emptyset, RQL = 0, GQ = \emptyset \; et \; wi = 1 \; \forall i.$

III.4. Evaluation de la qualité de service

Ce module a pour objectif d'évaluer la qualité de service afin de vérifier la satisfaction des exigences locales ou globales de l'utilisateur de qualité de service.

III.4.1. Evaluation de la qualité de service locale

Ce module permet d'évaluer la qualité de service de chaque service concret, en fonction des valeurs de paramètres de qualité et en considérant les préférences de l'utilisateur. Ainsi, les services concrets du même service abstrait sont classés en fonction de leurs QoS. Soit AS un service abstrait qui possède N services concrètes ; l'évaluation de la qualité de service au niveau local dans notre approche s'effectue en six étapes essentielles.

a) Première étape

Cette étape consiste à classifier les valeurs des différents paramètres de qualité selon les deux catégories ***SQP*** et ***DQP*** pour les N services concrets (tableau III.1).

Une telle classification est justifiée par le fait que les valeurs de ***SQP*** changent rarement, contrairement aux valeurs de ***DQP*** qui changent fréquemment. Par conséquent, le classement d'un service est mis à jour à chaque fois qu'il est invoqué. Cette classification a pour but d'optimiser la mise à jour des valeurs de paramètres de qualité, en ne mettant à jour que les paramètres dynamiques à chaque invocation. Dans le tableau III.1, $v_{i,j}$ est la valeur du paramètre de qualité j pour le service i.

	SQP_1	SQP_2	……	SQP_k	EL	RT	RE
Service 1	$V_{1,1}$	$V_{1,2}$	……	$V_{1,k}$	$V_{1,k+1,t}$	$V_{1,k+2,t}$	$V_{1,k+3,t}$
Service 2	$V_{2,1}$	$V_{2,2}$	……	$V_{2,k}$	$V_{2,k+1,t}$	$V_{2,k+2,t}$	$V_{2,k+3,t}$
…	……	……	……	……	……	……	……
Service N	$V_{n,1}$	$V_{n,2}$	……	$V_{n,k}$	$V_{n,k+1,t}$	$V_{n,k+2,t}$	$V_{n,k+3,t}$

Tableau III.1. Classification des paramètres.

b) Deuxième étape

Cette étape consiste à calculer la borne maximale de chaque paramètre de qualité en utilisant l'ensemble des formules suivantes décrites dans le tableau III.2.

Paramètre	Formule	Exemple
Les paramètres exprimés en pourcentage	max = 100 %	*EL*
Les paramètres exprimés entre 0 et 1	max = 1	*RE*
Les paramètres exprimés sur une échelle définie	max = la valeur maximale de l'échelle	Sécurité
Pour le reste des paramètres	max (paramètre j) = max (Vi,j), i=1... N.	cout

Tableau III.2. Calcul des bornes maximales, formules (14).

Cette étape est illustrée avec un exemple dans le tableau III.3.

Service	Coût	Sécurité	EL	RE	RT
S1	10	4	20	0,90	2
S2	150	4	30	1	8
S3	100	5	40	1	1
S4	170	3	51	0,60	1
Max	170	10	100	1	8

Tableau III.3. Exemple de calcul de borne max des paramètres de QoS.

c) Troisième étape

Dans cette étape, on calcule l'écart ($EV_{i,j}$) entre chaque la valeur de paramètre ($v_{i,j}$) et son maximum $Max(Parametre_j)$, selon la formule suivante.

$$EV_{i,j} = Max(Parametre_j) - v_{i,j} \qquad (15)$$

Par exemple, $\mathbf{EV_{1,1} = Max(prametre\ _1) - V\ _{1,1}}$ = MaxPrametre(Cout) – $V_{S1,cout}$ = 170 – 10

$\mathbf{EV_{1,1}}$ = 160.

Le tableau III.4 illustre le résultat de l'application de cette étape aux valeurs du **tableau III.3**.

Service	Ecart_coût	Ecart_Sécurité	Ecart_EL	Ecart_RE	Ecart_RT
S1	160	6	80	0,1	6
S2	20	6	70	0	0
S3	70	5	60	0	7
S4	0	7	49	0,4	7
Max	170	10	100	1	8

Tableau III.4. Exemple de calcul des écarts de valeurs des paramètres

d) Quatrième étape

Cette étape consiste à mettre en échelle l'écart des valeurs des paramètres ($\boldsymbol{EV_{i,j}}$) en les exprimant sous forme d'un écart compris entre 0 et 1($\mathbf{E_{i,j}}$).

Selon les contraintes liées aux paramètres de QoS, deux cas peuvent se présenter :

- **Minimisation des paramètres** (exemple : coût, EL, RT) **:**

$$\boldsymbol{E_{i,j} = \frac{EV_{i,j}}{Max(Parametre_{\,i})}} \qquad (16)$$

- **Maximisation des paramètres** (exemple : Sécurité, RE) **:**

$$\boldsymbol{E_{i,j} = 1 - \frac{EV_{i,j}}{Max(Parametre_{\,i})}} \qquad (17)$$

Le **tableau III.5** illustre l'application de cette étape aux valeurs du tableau III.4.

$\boldsymbol{E\ i,j}$	Ecart_cout	Ecart_Sécurité	Ecart_EL	Ecart_RE	Ecart_RT
S1	0,94	0,4	0,8	0,9	0,75
S2	0,11	0,4	0,7	1	0
S3	0,41	0,5	0,6	1	0,87
S4	0	0,3	0,49	0,6	0,87
Max	170	10	100	1	8

Tableau III.5. Exemple de mise à l'échelle des valeurs des paramètres

e) Cinquième étape

Cette étape a pour objectif d'évaluer la qualité de chaque service concret en fonction des écarts des valeurs des paramètres de qualité mises à l'échelle ($\mathbf{E_{i,j}}$), et des poids Wi fournis par l'utilisateur.

Compte tenu des deux types de paramètres de qualité, statiques et dynamiques, la qualité de service ***QoS*** est évaluée en fonction de la qualité de service statique (***SQoS)*** et de la qualité de service dynamique (***DQoS***) en prenant en considération les préférences de l'utilisateur **α**, **β** associées à ces deux types de qualité. Soit :

$$QoS_{i,t}(cs) = \frac{\alpha * SQoS(cs) + \beta * DQoS_t(cs)}{\alpha + \beta} \qquad (20)$$

✓ **SQoS** :

L'évaluation de la qualité de service statique d'un service ***i*** est réalisée selon la formule suivante :

$$SQoS_i(cs) = \frac{\sum_{j=1}^{j=k} W_j * (E_{i,j})}{\sum_{j=1}^{j=k} W_j} \qquad (18)$$

Où "**K"** représente le nombre de paramètres de qualité statiques.

✓ **DQoS :**

L'évaluation de la qualité de service dynamique d'un service ***i***, à l'instant **t**, est réalisée selon la formule suivante :

$$DQoS_{i,t}(cs) = \frac{\sum_{j=k}^{j=k+3} W_j * (E_{i,j})}{\sum_{j=k}^{j=k+3} W_j} \qquad (19)$$

	Ecart_ coût	Ecart_ Sécurité	Ecart_EL	Ecart_RE	Ecart_RT	SQoS	DQoS	QoS
Service 1	$E_{1,1}$	$E_{1,2}$	……	$E_{1,k+2}$	$E_{1,k+3,t}$	$SQoS_1$	$DQoS_1$	QoS_1
Service 2	$E_{2,1}$	$E_{2,2}$	……	$E_{2,k+2}$	$E_{2,k+3,t}$	$SQoS_2$	$DQoS_2$	QoS_2
…	……	……	……	……	……	…….	…….	…….
Service N	$E_{n,1}$	$E_{n,2}$	……	$E_{n,k+2,t}$	$E_{n,k+3,t}$	$SQoS_N$	$DQoS_N$	QoS_N
Poids W_j	W_1	W_2	……	W_{k+2}	W_{k+3}	WSQoS	WDQoS	

Tableau III.6. Evaluation de la QoS.

Le **tableau III.7** contient les résultats de l'application de cette étape aux valeurs du **tableau III.5**.

VALEUR	Ecart_coût	Ecart_Sécurité	Ecart_EL	Ecart_RE	Ecart_RT	SQoS	DQoS	QoS
S1	0,94	0,4	0,8	0,9	0,75	0,61	0,83	0,72
S2	0,11	0,4	0,7	1	0	0,28	0,57	0,42
S3	0,41	0,5	0,6	1	0,87	0,46	0,90	0,68
S4	0	0,3	0,49	0,6	0,87	0,18	0,69	0,43
w j	4	6	1	5	4	1	1	

Tableau III.7. Exemple d'évaluation de la QoS.

f) Sixième étape

Cette étape consiste à classer les services concrets cs en fonction de leur qualité de service QoS selon un ordre décroissant. En appliquant un classement sur les services du **tableau III.7**, on obtient le tableau III.8.

VALEUR	Ecart_coût	Ecart_Sécurité	Ecart_EL	Ecart_RE	Ecart_RT	SQoS	DQoS	QoS	classement
S1	0,94	0,4	0,8	0,9	0,75	0,61	0,83	0,72	1
S2	0,11	0,4	0,7	1	0	0,28	0,57	0,42	4
S3	0,41	0,5	0,6	1	0,87	0,46	0,90	0,68	2
S4	0	0,3	0,49	0,6	0,87	0,18	0,69	0,43	3
w j	4	6	1	5	4	1	1		

Tableau III.8. Exemple de classement des CS selon la QoS.

Ainsi, on obtient le classement suivant : S1 -> S3 -> S4 -> S2.

Les six étapes décrites précédemment sont exécutées par l'algorithme de classement des CS, suivant :

Algorithme 1 : Concrete Service Classification (CSClsf)

Inputs: Abstract Service (*AS)*
Outputs: Abstract Service (*AS) with CS classified.*
Begin
1. **For each** cs in AS **do**
2. **For each** SQP in CS **do**
3. Use values of SQP and Compute the MAX using Formulas (14)
4. Compute the gap using Formula (15)
5. Compute the new gap between 0 and 1, using Formulas (16, 17)
6. **End for**
7. Use Wi and Compute $SQoS$ using Formula (18).
8. **For each** DQP in CS **do**
9. Use values of DQP and Compute the MAX using Formulas (14)
10. Compute the gap using Formula (15)
11. Compute the new gap between 0 and 1, using Formula (16) or Formula (17)
12. **End for**
13. Compute $DQoS$ using Formula (19).
14. Compute QoS using Formula (20).
15. **End for**
16. Classify CS in AS using QoS with descending order.

End

III.4.2. Evaluation de la qualité de service globale

Après avoir exposé notre approche concernant l'évaluation de la QoS d'un service concret atomique CS, nous présentons notre approche concernant l'évaluation de la QoS d'un service composite. Cette approche a comme objectif de vérifier la satisfaction des différentes contraintes globales spécifiées par l'utilisateur afin de sélectionner le meilleur service composite.

L'évaluation de la QoS composite varie selon les paramètres considérés et selon le modèle du service abstrait dans le schéma de composition. Dans notre approche, nous distinguons trois modèles possibles: séquentiel, parallèle et boucle (figure III.4).

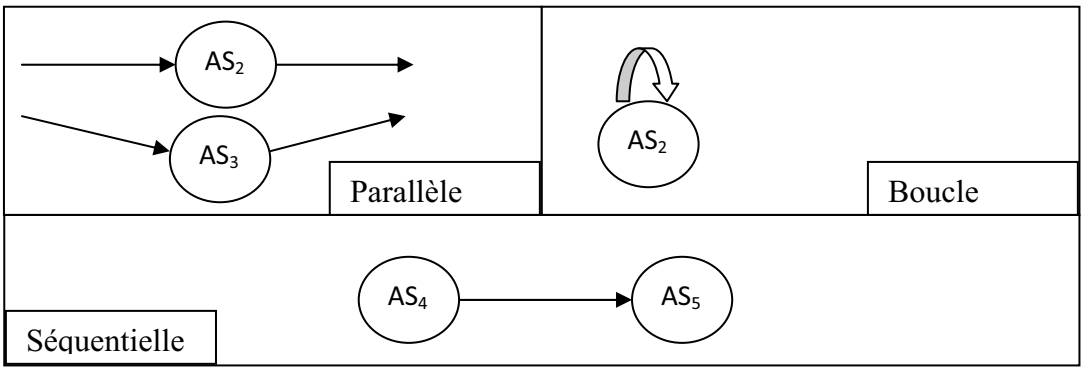


Figure III.4. Position d'un service dans le schéma de composition.

Selon le modèle du service abstrait dans le schéma de composition, nous associons à chaque paramètre de qualité considéré une métrique, comme le montre le tableau III.9 :

	coût	**Sécurité**	**EL**	**RE**	**RT**	**AV**
séquence (a,b)	coût(a)+ coût(b)	min (sécu(a), sécu (b))	$\frac{\text{ELa} + \text{ELb}}{2}$	REa * REb	RTa + RTb	AVa*Avb
parallèle (a,b)	coût(a)+ coût(b)	min (sécu (a), sécu (b))	min (ELa,ELb)	$\frac{\text{REa} + \text{REb}}{2}$	max (RTa,RTb)	AVa*AVb
boucle (n fois)	$n * \text{coût(a)}$	sécurité (a)	$\frac{\sum_{i=1}^{i=n}(\text{ELi})}{n}$	$\prod_{i=1}^{i=n} REi$	$\sum_{i=1}^{i=n}(\text{RTi})$	$\prod_{i=1}^{i=n} AVi$

Tableau III.9. Métriques d'évaluation de la QoS d'un service composite

A titre illustratif, on considère le service composite représenté par le graphe suivant :

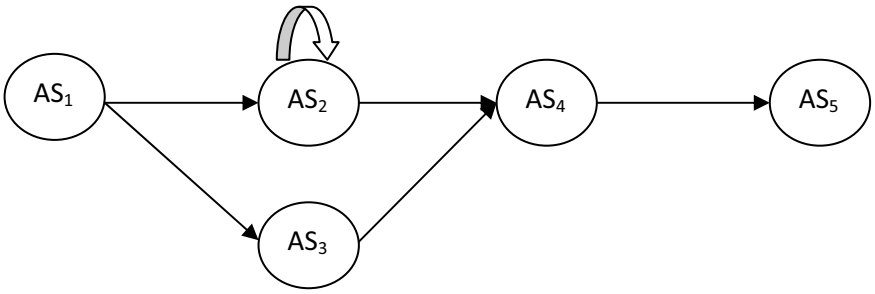


Figure III.5. Exemple de graphe d'un service composite.

Ce service composite est décrit en utilisant l'expression régulière suivante :

$$séquence(AS_1, \text{parallèle (boucle } (\text{AS}_2), \text{AS}_3), \text{AS}_4, AS_5)$$

L'évaluation des différents paramètres de QoS pour le service composite considéré dans l'exemple précédent, est décrite dans le tableau III.10.

	$séquence(AS_1, \text{parallèle (boucle } (\text{AS}_2), \text{AS}_3), \text{AS}_4, AS_5)$
Coût	$coût(AS_1) + n * \text{coût } (\text{AS}_2) + \text{coût}(\text{ AS}_3) + \text{coût}(\text{ AS}_4) + \text{ coût}(\text{ } AS_5)$
Sécurité	$Min\ (\text{Sécurité}(AS_1), \text{Sécurité}(\text{AS}_2), \text{Sécurité}(\text{ AS}_3), \text{Sécurité}(\text{ AS}_4), \text{Sécurité}(\text{ } AS_5))$
EL	$\frac{EL(AS_1) + Min(\frac{\sum_{i=1}^{i=n} ELi\ (\text{AS}_2)}{n}, \text{EL}(\text{ AS}_3)) + \text{EL}(\text{ AS}_4) + \text{ EL}(\text{ } AS_5)}{4}$
RE	$RE(AS_1) * \frac{\prod_{i=1}^{i=n} REi(AS_2) + \text{RE}(\text{ AS}_3)}{2} * \text{RE}(\text{ AS}_4) * \text{ RE}(\text{ } AS_5)$
RT	$RT(AS_1) + Max(\sum_{i=1}^{i=n} \text{RT}(\text{AS}_2), \text{RT}(\text{ AS}_3)) + \text{RT}(\text{ AS}_4) + \text{ RT}(\text{ } AS_5)$
AV	$AV(AS_1) * \prod_{i=1}^{i=n} AVi\ (AS_2) * \text{AV}(\text{ AS}_3) * \text{AV}(\text{ AS}_4) * \text{ AV}(\text{ } AS_5)$

Tableau III.10. Exemple d'évaluation des paramètres de QoS d'un service composite.

III.5. La sélection de service

Dans notre approche, nous considérons deux types de sélections, une sélection locale, appliquée à chaque service abstrait AS, et une sélection globale appliquée sur l'ensemble des services abstraits constituant le service composite (Figure III.6).

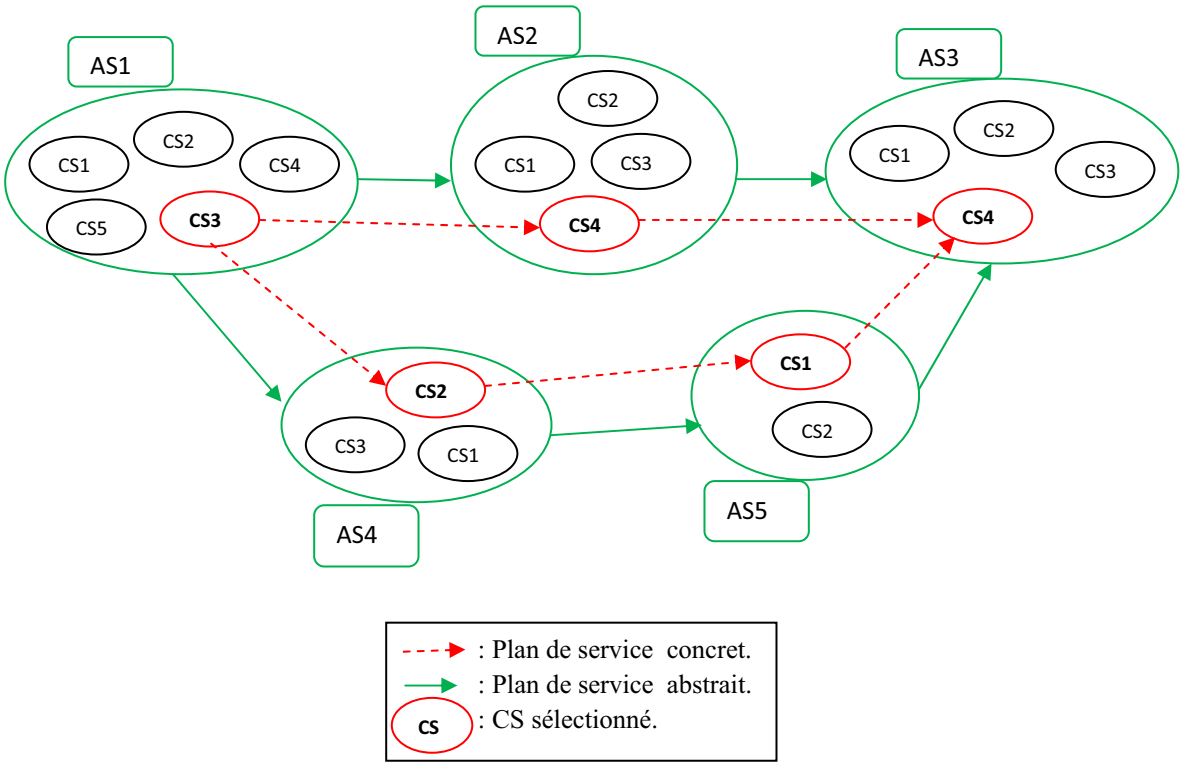


Figure III.6. Exemple de sélection d'un service.

III.5.1. La sélection de service locale

La sélection de service locale consiste à choisir un service concret CS dans un ensemble de services contenus dans un service abstrait AS. Cette sélection est effectuée en choisissant le meilleur service concret en termes de QoS dans un sous-ensemble de services concrets répondant à MQ (spécifique et générale) et satisfaisant RQL. Ce sous-ensemble noté SatCS (*Satisfied Concrete Services*), est calculé en utilisant la formule suivante :

$$\text{SatCS} = \text{cs} \in \text{AS} / \begin{Bmatrix} \text{cs satisfies } \text{MQ}_{\text{général}} \cup \text{MQ}_{\text{spécifque}} \\ \text{QoS}_t(\text{cs}) \geq \text{RQL} \end{Bmatrix} \quad (21)$$

La procédure complète de sélection d'un service concret est réalisée à l'aide de l'algorithme suivant :

Algorithme 2, Concrete Service Selection (CSS)

Inputs: Abstract Service (*AS)***,** Satisfied Inputs (SI), Desired Parameters (DP)
Begin
1. response=false ;
2. *SatCS = the set of cs contained in AS*
3. **While** ((response==false**) and** *(SatCS* ≠∅)) **do**
4. **CSClsf** (AS, AS_classified);
5. Compute *SatCS* using formula (21);
6. **If** (*SatCS* ≠∅) **do**
7. Select the first *cs* according to the rank in *SatCS*
8. **CSI** (cs, DP, SI; response, outputs)
9. **End iF**
10. **End while**
11. **If** (response=true)
12. Send outputs to the service monitoring module
13. **Else**
14. "There is no achieved cs due to the environment context"
15. **End if**

End

Une telle sélection a comme avantage de réduire le nombre de CS à parcourir avant de trouver le meilleur service composite et par conséquent, de réduire le temps de la recherche. Cependant, elle ne tient pas compte des contraintes globales.

III.5.2. La sélection de service globale

La sélection de service globale consiste à satisfaire les exigences de l'utilisateur spécifiées sur le service composite, après une sélection locale effectuée sur chaque service abstrait AS. L'objectif du module de sélection globale est de sélectionner le meilleur service composite parmi toutes les combinaisons possibles de services concrets.
Le nombre total des services composites est calculé à partir de la formule suivante :

$$NbreServiceComposite = \prod_{i=1}^{i=n} |AS_i| \quad telle\ que\ |AS_i| = nombre\ de\ CS\ dans\ AS_i$$

Malgré les avantages présentés par la sélection locale en terme de temps de calcul, cette solution ne garantie pas la satisfaction des contraintes globales définies par l'utilisateur. Pour illustrer cet inconvénient, considérons un service composite représenté par le schéma suivant :

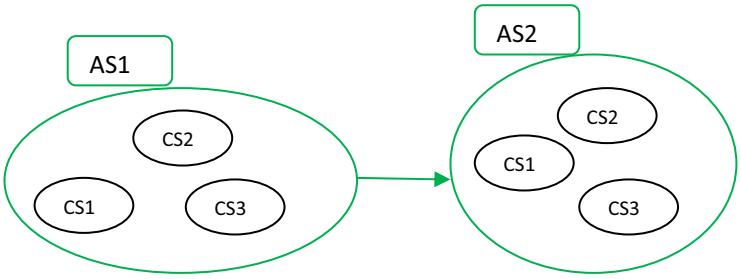


Figure III.7. Exemple de sélection globale.

L'utilisateur définit une seule contrainte globale : $Cout_{global} < 150\ euros$. Après l'évaluation locale en fonction de la QoS des services AS1 et AS2, nous obtenons le classement suivant :

$AS_1 = \{CS_1, CS_3, CS_2\}$
$AS_2 = \{CS_3, CS_1, CS_2\}$
Cependant, les valeurs du paramètre coût de chaque service concret CS sont :
$AS_1 = \{CS_1(coût = 145), CS_3(coût = 30), CS_2(coût = 90)\}$
$AS_2 = \{CS_3(coût = 10), CS_1(coût = 30), CS_2(coût = 50)\}$

Il est clair que, choisir CS_1 de AS_1 et CS_3 de AS_2 viole la contrainte $Coût_{global} < 150\ euros$, malgré que CS_1 de AS_1 et CS_3 de AS_2 possèdent la QoS la plus élevée. C'est pour cette raison que le simple choix des premiers CS de chaque AS est insuffisant et qu'il est nécessaire de définir un mécanisme qui parcourt les CS de chaque AS jusqu'à l'obtention du meilleur service composite respectant toutes les contraintes de composition définies par l'utilisateur. Afin de résoudre ce problème, nous avons opté dans notre approche pour une formulation du problème en problème de satisfaction de contraintes (CSP pour Constraint Satisfaction Problem).

III.5.2.1. Modélisation avec CSP

a) Les variables

Les variables sont les différents services abstraits AS_i, et chaque AS a comme domaine de définition l'ensemble des services concrets CS qu'il contient. c.à.d: **$D(AS_i) = \{ cs \ / \ cs \in AS_i\}$.**

b) Les contraintes

On considère dans notre approche deux types de contraintes :

✓ **Les contraintes concrètes**

Ce type de contraintes représente les différentes exigences de l'utilisateur liées au service composite, comme par exemple coût $_{global}$ < 150 Euros.

✓ **Les contraintes abstraites**

Les contraintes abstraites sont les contraintes associées aux paramètres de qualité (coût, EL, etc.) et au modèle du service abstrait (parallèle, séquentielle et boucle) dans le schéma de composition, ce qui permet de choisir la métrique (décrite dans le **tableau III.9**) à appliquer pour évaluer le service composite.

c) Le parcours du graphe de CSP

Le parcours du graphe est réalisé selon la méthode du **Backtracking**. Afin de minimiser le nombre de nœuds (CS) à parcourir dans le graphe, et par conséquent le temps de parcours induit par cette méthode, nous avons utilisé des heuristiques sur l'ordre de parcours des AS et des CS.

✓ **L'ordre de parcours des variables (AS_i)**

L'ordre de parcours de variables est défini en utilisant l'heuristique $\frac{Dom}{Deg}$ telle que :

Dom est le cardinal des CS dans AS, c.à.d, $Dom_i = |AS_i|$.

Deg est le degré d'implication d'une variable AS dans les contraintes, en d'autres termes, il s'agit du nombre de voisins liés avec des contraintes à la variable *ASi.*

Les variables sont ordonnées selon l'ordre croissant de leurs $\frac{Dom}{Deg}$, tel que le service abstrait AS qui a le plus petit domaine avec le plus grand nombre de voisins est classé en premier (Algorithme 3).

Algorithme 3, Classification des AS (AS classification)

Inputs: ensemble de AS.
Outputs: Ensemble de AS ordonné.
Begin
1. **For each** AS do
2. $score_i = \frac{Dom(AS_i)}{Deg(AS_i)}$
3. Insérer_ordonner (AS_i , $score_i$, ensemble_de_AS)
4. **End for**

End

✓ **L'ordre de parcours des valeurs (CSj)**

L'ordre de parcours des CS dans chaque AS s'effectue selon le classement des services concrets déjà établi dans la phase d'évaluation locale.

III.5.2.2. Procédure de sélection globale d'un service composite

Avant d'exposer la procédure complète de sélection globale d'un service composite, nous présentons tout d'abord l'algorithme proposé pour la sélection des CS dans l'ensemble des AS pour la sélection composite **CSSCom**. Cet algorithme effectue pour chaque AS un classement en utilisant l'algorithme **CSClsf** que nous avons déjà présenté, et détermine l'ensemble des CS (*SatCS*) qui satisfont les exigences de l'utilisateur définies sur chaque AS (Algorithme 4).

Algorithme 4, Concrete Service Selection Composite (CSSCom)

Inputs: Vector of Abstract Services (*AS)*, Satisfied Inputs (SI), Desired Parameters (DP)
Outputs: Vector of Abstract Services (*AS) with CS classified*, Is_composite

Begin
1. Is_composite = true;
2. **For each** AS **do**
3. **CSClsf** (AS, AS_classified);
4. Compute *SatCS* using formula (21);
5. **If** (*SatCS* == ∅)
6. Is_composite = false;
7. "There is no a service composite"
8. Return (Is_composite);
9. **End if**
10. **End For**
11. **End**

La sélection de service globale se fait en utilisant la méthode de parcours Backtracking, et en appliquant les deux heuristiques définies sur l'ordre des variables et des valeurs (Algorithme 5).

Algorithme 5, Composite Service Selection (ComSS)

```
Inputs:    Tab_AS :Vector of Abstract Services (AS),
           tab_param : Vector of Parameters of composite constraints
Outputs: Tab_AS_instenced : Vector of Abstract Services (AS) instanced with CS selected
 Begin
1.   Is_composite = false;
2.   CSSCom ( tab_AS, SI, DP, tab_AS, Is_composite ) ;    //Figure 11
3.   If (Is_composite == false) {
4.       Return (false) ;
5.   }
6.   AS_classification (tab_AS_NoOrdre, tab_AS );  //Figure 10
7.
8.   Tab_AS_instenced = Ø;
9.
10.  while (Tab_AS ≠ Ø ) do
11.       CS = CS_chosed to explore;
12.       While (    AS.set(SC) ≠ Ø && Verifier_Contraintes_composite( CS , tab_AS_ instenced) == false ) do
13.            CS = CS_chosed to explore;
14.       End while
15.       If (AS.set(SC) ≠ Ø ) {
16.            AS.CS_chosed = CS ;
17.            tab_AS_ instenced.add (AS) ;
18.            If (tab_AS == Ø) {
19.                 return true;
20.            }
21.       }else{
22.            While (  AS.set(SC) ==  Ø  ) do
23.                 Backtracking ;
24.            End while
25.            if(AS(0) .set(SC)  == Ø) {
26.                 return( false );
27.            }
28.       }
29.  End while
30.  return false;
End
```

III.5.2.3. Procédure de traitement des défaillances dans un service composite

Une fois le service composite sélectionné, son invocation est effectuée. Il s'agit d'invoquer chaque service concret CS sélectionné dans chaque AS, et en cas d'échec, la procédure de sélection de service composite est à nouveau exécutée à partir du AS dont l'invocation de l'un des CS sélectionné a échouée (Algorithme 6).

Algorithme 6, **Composite Service Invocation (ComSerInvocation)**

Inputs: Vector of Abstract Services (Tab_*AS)*

Begin

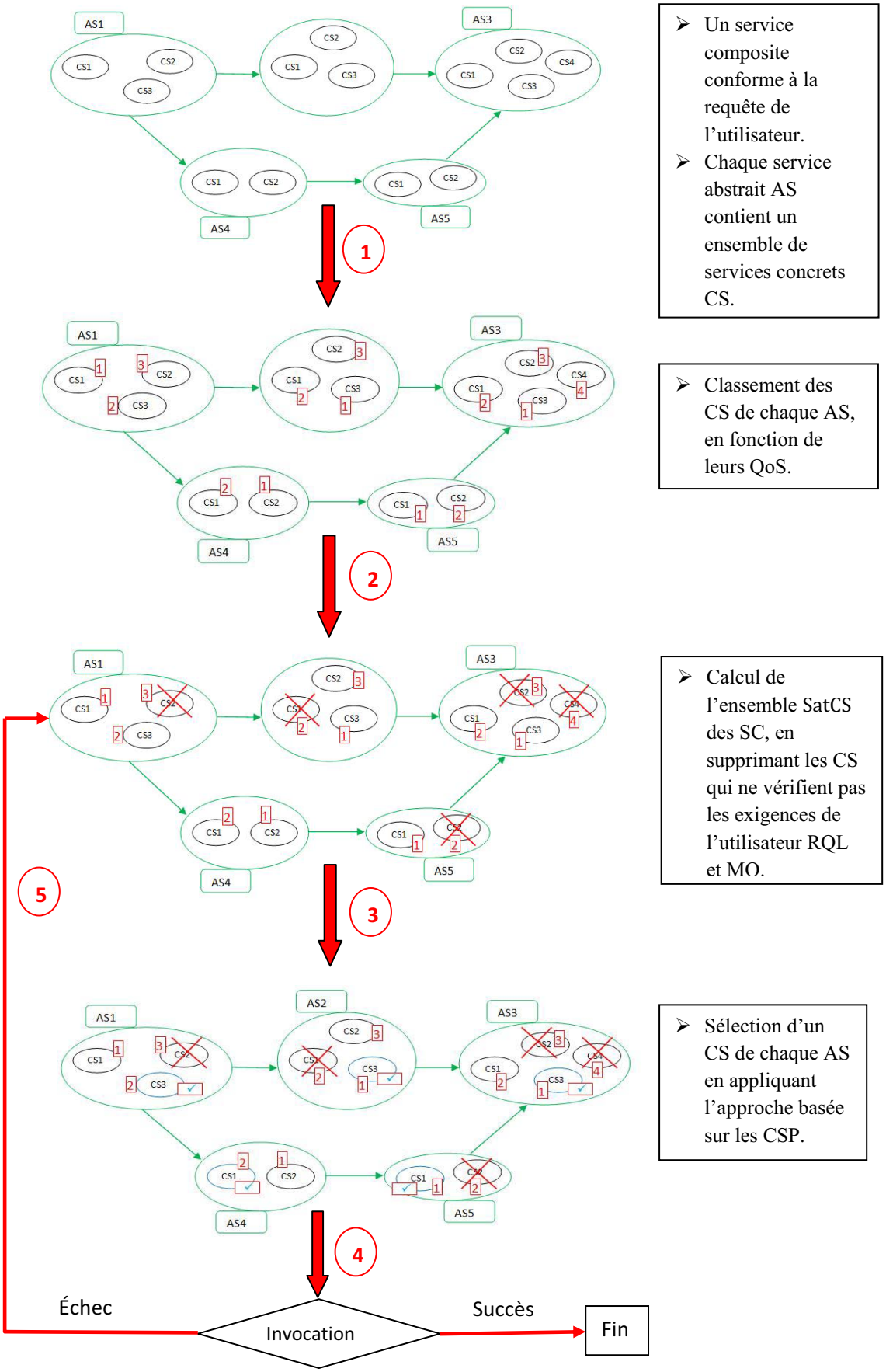
```
1.  CSR=false
2.  For each AS  do
3.          CSR = CSI (AS.cs_selected, CSR);
4.          If (CSR == false)
5.                  Composite Service Selection (Tab_AS);
6.          Else
7.                  Remove AS from Tab_AS ;
8.           End if
9.  End For
10. End
```

III.5.2.4. Schéma de sélection globale d'un service composite

Le schéma suivant illustre la procédure complète de sélection de service composite. Les différentes étapes de ce schéma, sont expliquées dans le tableau III.11.

Etape 1	Evaluer la QoS de chaque CS et effectuer un classement des CS selon leurs QoS.
Etape 2	Vérifier les exigences de l'utilisateur et supprimer les CS qui ne vérifient pas RQL et MQ
Etape 3	Sélectionner un CS par AS, en appliquant l'approche de sélection globale de service composite basée sur le CSP.
Etape 4	Invoquer chaque CS sélectionné de chaque AS, selon le plan de composition.
Etape 5	En cas d'échec, la procédure de sélection de service composite est à nouveau exécutée, à partir du AS dont l'invocation de l'un des CS sélectionné a échouée.

Tableau III.11. Etapes de sélection globale d'un service composite.



- Un service composite conforme à la requête de l'utilisateur.
- Chaque service abstrait AS contient un ensemble de services concrets CS.

- Classement des CS de chaque AS, en fonction de leurs QoS.

- Calcul de l'ensemble SatCS des SC, en supprimant les CS qui ne vérifient pas les exigences de l'utilisateur RQL et MO.

- Sélection d'un CS de chaque AS en appliquant l'approche basée sur les CSP.

Figure III.8. Schéma de sélection globale d'un service composite.

III.6. Invocation des services concrets (l'exécution)

Ce module a pour objectif d'invoquer un service concret sélectionné par le module de sélection de service. Il fait appel au module de mesure (voir section III.7) pour évaluer et mettre à jour les différents paramètres de qualité. Le résultat de l'invocation peut être un succès ou un échec. Nous distinguons deux modes d'invocation selon l'approche de mesure considérée.

III.6.1. Invocation basée sur l'approche sans mémoire

Dans ce mode d'invocation, seules les valeurs de paramètres de qualité à l'instant t sont considérées. En effet, avant de lancer l'exécution du service concret, une vérification de sa disponibilité est effectuée. Le service est ensuite invoqué tout en respectant les contraintes liées au timeout et à l'énergie. A la fin de ce processus, les valeurs de paramètres de qualité sont évaluées et mises à jour, et un état d'exécution pouvant être un succès ou un échec est retourné. La procédure d'invocation complète de l'approche sans mémoire est décrite dans l'algorithme 7 ci-dessous.

Algorithme 7: Concrete Service Invocation (CSI)

Inputs: concrete service (*cs)*, Desired Parameters (DP)
Outputs: concrete service response (CSR), obtained cs outputs
Begin
1. CSR=false
2. AV(cs)=Check availability (cs);
3. **If** (AV(cs)≠0) **then**
4. Establish connection with cs
5. Send inputs to cs
6. **While** (timeout(cs) is not expired and DP are not received and
7. ($EL(CS)_{requis}$ <= EL(Device(cs)))) **do**
8. Wait and Receive outputs from cs
9. **End while**
10. Compute and Update RT(CS) with formulas (0; 1; 7)
11. Compute and Update RE(CS) with formula 2
12. Compute and Update EL(CS) with formulas (3; 4)
13. **End if**
14. **If** (RE(cs)==1) **then**
15. CSR=true
16. **End if**

End

III.6.2. Invocation basée sur l'approche avec mémoire

Dans ce mode, les valeurs moyennes des paramètres de qualité en fonction de nombre d'invocations de services sont considérées. La procédure d'invocation de l'approche avec mémoire est décrite dans l'algorithme 8.

Algorithme 8: Concrete Service Invocation (CSI)

Inputs: concrete service (*cs)*, Desired Parameters (DP), cs inputs
Outputs: concrete service response (CSR), obtained cs outputs
Begin
1. CSR=false;
2. AV(cs)=Check availability (cs);
3. **If** (AV(cs)≠0) **then**
4. Establish connection with cs
5. $InvocationCounter$ ++
6. Send inputs to cs
7. **While** (timeout(cs) is not expired and DP are not received and
8. $(\boldsymbol{EL}(\boldsymbol{CS})_{\boldsymbol{consommer}}$ <= EL(Device(cs)))) **do**
9. Wait and Receive outputs from cs
10. **End while**
11. Compute and Update RT(CS) with formulas (0; 1; 7; 8)
12. Compute and Update RE(CS) with formulas (2; 9)
13. Compute and Update EL(CS) with formulas (3; 4 ; 10)
14. **End if**
15.
16. **If** (RE(cs)==1) **then**
17. CSR=true ;
18. $Cpt\_succuful$++ ;
19. **End if**
20. Compute and Update AV(CS) with formula (12)

End

III.7. Les mesures de paramètres de qualité Dynamiques (*DQP)*

Comme nous l'avons déjà expliqué précédemment, les paramètres dynamiques dépendent principalement du contexte d'utilisation de service, et cela a un impact immédiat sur la qualité du service. C'est pourquoi nous proposons un module ayant pour objectif la mesure et la mise à jour des différents paramètres de qualité de services dynamiques.

Dans ce module, nous proposons deux types d'approches : les approches sans mémoire et les approches avec mémoire.

III.7.1. Approches sans mémoire

Les mesures de paramètres de qualité, dans ce type d'approches, ne dépendent pas de l'historique de leurs valeurs. Ces mesures sont calculées en fonction des valeurs de paramètres de qualité à l'instant t. Les mesures de différents paramètres de qualité sont calculées selon les formules suivantes :

a) ***Disponibilité (AV)***

Un service peut être disponible (c.à.d. AV = 1) ou indisponible (c.à.d. AV = 0), et cela peut être engendrée par plusieurs raisons, à savoir : des pannes du réseau, l'échec d'un service, la modification ou la configuration d'un service, etc.

Nous distinguons deux cas où l'indisponibilité d'un service peut se produire : avant l'invocation ou pendant l'exécution.

b) ***Temps de réponse (RT)***

L'invocation de service est contrôlée par un délai (**timeout**) spécifié pour chaque service concret **CS** en fonction de son temps de réponse (**RT**) comme suit:

$$\boldsymbol{timeout(cs) = \alpha . RT(cs)} \textbf{ avec } \boldsymbol{\alpha > 1} \qquad (\mathbf{0})$$

Dans le cas où le timeout est écoulé sans avoir une réponse du service sollicité, le facteur $\boldsymbol{\alpha}$ est mis à jour de façon à avoir un timeout plus élevé en appliquant la formule suivante :

$$\boldsymbol{\alpha} = \begin{cases} \boldsymbol{\alpha} * \boldsymbol{\beta} \text{ , Telle que } \boldsymbol{\beta > 1} \text{ si } (\boldsymbol{timeout(cs) = 0} \textit{ et } \boldsymbol{RE(cs) = 0}) \\ \boldsymbol{\alpha} \textbf{ Sinon} \end{cases} \qquad \textbf{(7)}$$

RT est la durée qui sépare le début de l'invocation ($\mathbf{T_{initial}}$) du service de l'instant de la réception ($\mathbf{T_{final}}$) de tous les paramètres désirées. Le temps de réponse à l'instant t est calculé par la formule suivante :

$$\mathbf{RT_t = T_{final} - T_{initial}} \qquad (1)$$

c) ***Fiabilité (RE)***

Soit $\boldsymbol{etat(p)}$ l'état perçu d'un paramètre **p** prenant une valeur comprise entre 0 et 1. La formule suivante permet le calcul de la fiabilité ***RE*** à l'instant t :

$$\mathbf{RE_t}(cs) = \frac{\boldsymbol{nbrOfReceivParam_t}\,(cs) - \sum_{i=1}^{i=n}(1 - \boldsymbol{etat(parametres_i)})}{\boldsymbol{nbrOfDesirParam_t}(cs)} \qquad (2)$$

Où :

$\boldsymbol{etat(parametres_i)} = v.\ telle\ que\ v \in [0\,..\,1]$

$\boldsymbol{nbrOfReceivParam_t}$: nombre de paramètres reçus.

$\boldsymbol{nbrOfDesirParam_t(cs)}$: nombre de paramètres souhaités.

RE = 1, si tous les paramètres souhaités sont reçus correctement ($\boldsymbol{etat(parametres_i)}$ =**1**) avant l'expiration du timeout spécifié. Si **RE = 1**, la réponse du service est considérée comme positive, sinon elle est négative.

d) ***Niveau d'énergie (EL)***

Le niveau d'énergie est calculé en utilisant la formule suivante :

$$\boldsymbol{EL_t\,(CS)} = \frac{\boldsymbol{EL_{requis(CS)_t}}}{\boldsymbol{EL_{Device(CS)_t}}} \qquad (3)$$

La mise à jour de la quantité d'énergie requise par le service est obtenue à partir de la formule suivante :

$$\boldsymbol{EL_{requis(CS)_{t+1}} = EL_{Device(CS)_t} - EL_{Device(CS)_{t+1}}} \qquad (4)$$

Où :

$\boldsymbol{EL_{requis(CS)_t}}$ **:** La quantité d'énergie à consommer pour la prestation de service à l'instant t.

$\boldsymbol{EL_{Device(CS)_t}}$ **:** Le niveau de la batterie du dispositif hébergeant le service à l'instant t (avant l'invocation).

$\boldsymbol{EL_{Device(CS)_t}}$ **:** Le niveau de la batterie du dispositif qui héberge le service à l'instant t+1 (après l'exécution).

Un service concret CS dont le niveau d'énergie du dispositif associé est nul ($\boldsymbol{EL_{Device(CS)_t} = 0}$) **o**u le niveau d'énergie requis par le service est supérieur au niveau de la batterie du dispositif ($\boldsymbol{EL_{requis(CS)_t} > EL_{Device(CS)_t}}$) ne sera pas pris en considération lors de la sélection.

III.7.2. Approches avec mémoire

Dans ce type d'approches, l'évolution des valeurs de paramètres de qualité dynamiques (*DQP)* au fil du temps est prise en compte. Pour ce faire, nous associons à chaque service concret un compteur ***InvocationCounter*(cs)** qui calcule le nombre de fois que le service (CS) a été invoqué.

InvocationCounter*(cs)** est initialisé à zéro, puis incrémenté d'une unité (InvocationCounter*(cs) + 1**) à chaque invocation.

Afin d'évaluer les différents paramètres de qualité, nous établissons de nouvelles formules basées sur celles de l'approche sans mémoire.

a) ***Disponibilité (AV)***

La disponibilité ***(AV)*** dans les approches avec mémoire indique le nombre de fois où un service concret est exécuté avec succès par rapport au nombre de ses invocations. Elle est calculée selon la formule suivante :

$$\boldsymbol{AV(cs) = \frac{Cpt\_succuful(cs)}{InvocationCounter(cs)}} \qquad (12)$$

Tel que : $\boldsymbol{Cpt\_succuful(cs)}$ est le nombre de fois où le CS est exécuté avec succès.

b) ***Temps de réponse (RT)***

Dans ce type d'approches avec mémoire, on considère le temps de réponse moyen calculé en fonction du temps de réponse à l'instant t (obtenu par les formules 0, 1 et 7) selon la formule suivante :

$$RT(\boldsymbol{cs}) = \frac{\sum_{t=1}^{t=InvocationCounter(cs)} \mathbf{RT}(\boldsymbol{cs})_{\mathbf{t}}}{\boldsymbol{InvocationCounter(cs)}} \qquad (8)$$

c) ***Fiabilité (RE)***

De même, pour la fiabilité (RE), nous considérons la fiabilité moyenne qui est calculée comme suit :

$$\boldsymbol{RE(cs)} = \frac{\sum_{t=1}^{t=InvocationCounter(cs)} \mathbf{RE(cs)_t}}{\boldsymbol{InvocationCounter(cs)}} \quad (9)$$

d) ***Niveau d'énergie (EL)***

Le taux d'énergie moyen (EL) est calculé en utilisant les formules 3 et 4 et la formule suivante :

$$\boldsymbol{EL(cs)} = \frac{\sum_{t=1}^{t=InvocationCounter(cs)} \mathbf{EL(cs)_t}}{\boldsymbol{InvocationCounter(cs)}} \quad (10)$$

De la même manière, on peut adapter la formule (10) afin de l'utiliser pour évaluer d'autres paramètres de qualité dynamiques, comme par exemple le taux d'occupation de l'espace mémoire.

III.8. Conclusion

Nous avons présenté dans ce chapitre une approche de sélection de services.

Cette approche effectue une sélection globale basée sur la qualité de service en prenant en compte plusieurs facteurs, à savoir les facteurs liés aux exigences et préférences de l'utilisateur sur le plan local et global et les facteurs liés au contexte.

Dans le chapitre suivant, nous présentons la mise en œuvre en simulation de notre approche et analysons ses performances.

4

Implémentation et évaluation des performances de l'approche proposée

IV.1. Introduction

Dans ce chapitre, nous présentons tout d'abord les détails d'implémentation de l'approche proposée puis nous évaluons ses performances en termes de temps d'exécution. Une analyse comparative des performances de cette approche avec les approches existantes est également développée.

IV.2. Conception et implémentation

Pour l'évaluation de l'approche de sélection proposée, nous avons conçu et développé une application permettant d'une part, de simuler un système ubiquitaire et d'autre part, de sélectionner le meilleur service.

Les classes considérées dans la conception de l'application sont décrites dans le tableau IV.1.

Classe	Description
Decouvert_AS	Elle joue le rôle d'un annuaire. Elle contient les classes : Service abstrait, Paramètre de qualité composite et Paramètre de qualité config.
Service abstrait	Cette classe représente un service abstrait. Elle contient les classes : Service concret et Paramètre de qualité AS.
Service concret	Cette classe représente un service concret. Elle contient la classe Paramètre de qualité CS.
Paramètre de qualité AS	Cette classe contient les informations des paramètres de qualité concernant un service abstrait, comme par exemple : le nom du paramètre, la valeur maximale, le poids et la valeur exigée.
Paramètre de qualité CS	Cette classe contient les informations des paramètres de qualité concernant un service concret, comme par exemple le nom du paramètre et sa valeur.
Paramètre de qualité config	Cette classe représente les informations de configurations des paramètres de qualité concernant tous les services abstraits: nom du paramètre, type du paramètre (statique ou dynamique) et la contrainte liée à ce paramètre (à maximiser ou à minimiser).
Paramètre de qualité composite	Cette classe contient les informations des paramètres associées aux contraintes de composition (globale), comme par exemple le nom du paramètre et sa valeur globale exigée.
Sélection locale	Cette classe a pour objectif d'effectuer une sélection locale en utilisant la classe Decouvert_AS.
Sélection globale	Cette classe effectue une sélection globale en faisant appel aux classes : Sélection locale, CSP et Parsing expression régulière.
CSP	Cette classe implante l'approche CSP.
Parsing expression régulière	Cette classe sert à traiter les expressions régulières utilisées pour décrire le plan abstrait.

Tableau IV.1. Description des classes considérées dans la conception.

La structure globale de notre application est décrite dans le digramme de classes suivant :

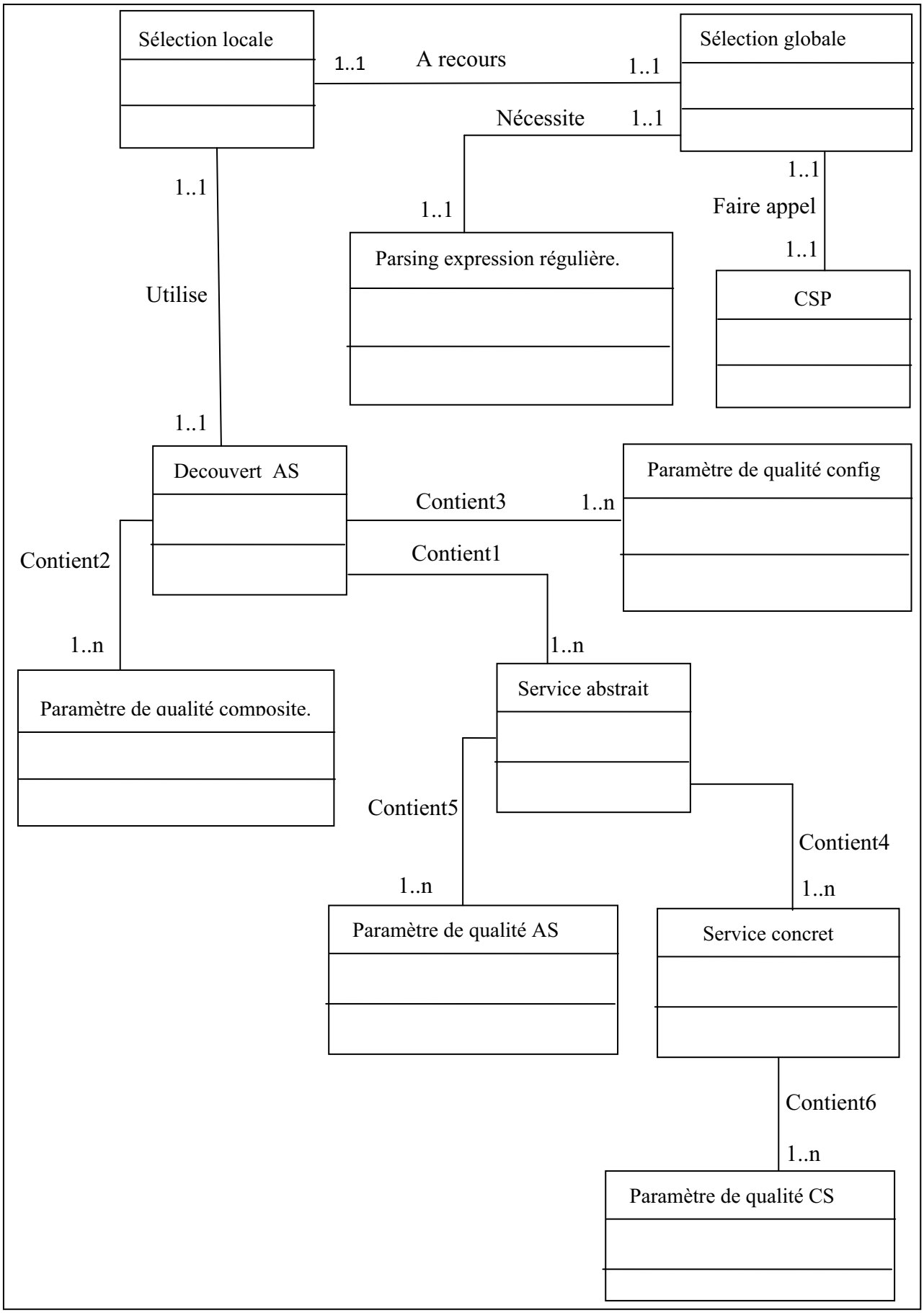


Figure IV.1. Diagramme de classes.

L'implémentation de notre application est réalisée en utilisant les langages « JAVA » et XML.

Le fichier XML est utilisé pour décrire le plan abstrait, comme le montre la figure IV.2 :

```xml
<?xml version="1.0" encoding="UTF-8"?>
<composition>
    <positionement>
        sequence(0,1,parallelle(2,3),4)
    </positionement>

    <dimention>5</dimention>

    <boucle>
        <AS id="0">1</AS>
        <AS id="1">1</AS>
        <AS id="2">1</AS>
        <AS id="3">1</AS>
        <AS id="4">1</AS>
    </boucle>

    <matrice>
        <AS id="0">1</AS>
        <AS id="1">2,3</AS>
        <AS id="2">4</AS>
        <AS id="3">4</AS>
        <AS id="4">-1</AS>
    </matrice>

</composition>
```

Figure IV.2. Exemple du plan abstrait en utilisant XML.

Le fichier contient l'expression régulière qui décrit la structure du plan abstrait, le nombre de services abstraits et les différents liens entre les services abstraits sous la forme d'une matrice.

IV.3. Fonctionnalités de l'application

Dans l'application développée, l'utilisateur a la possibilité de spécifier ses besoins, ses exigences et ses préférences, sur le plan local, de façon générale ou spécifique et sur le plan global, grâce à l'interface illustrée dans la figure IV.3.

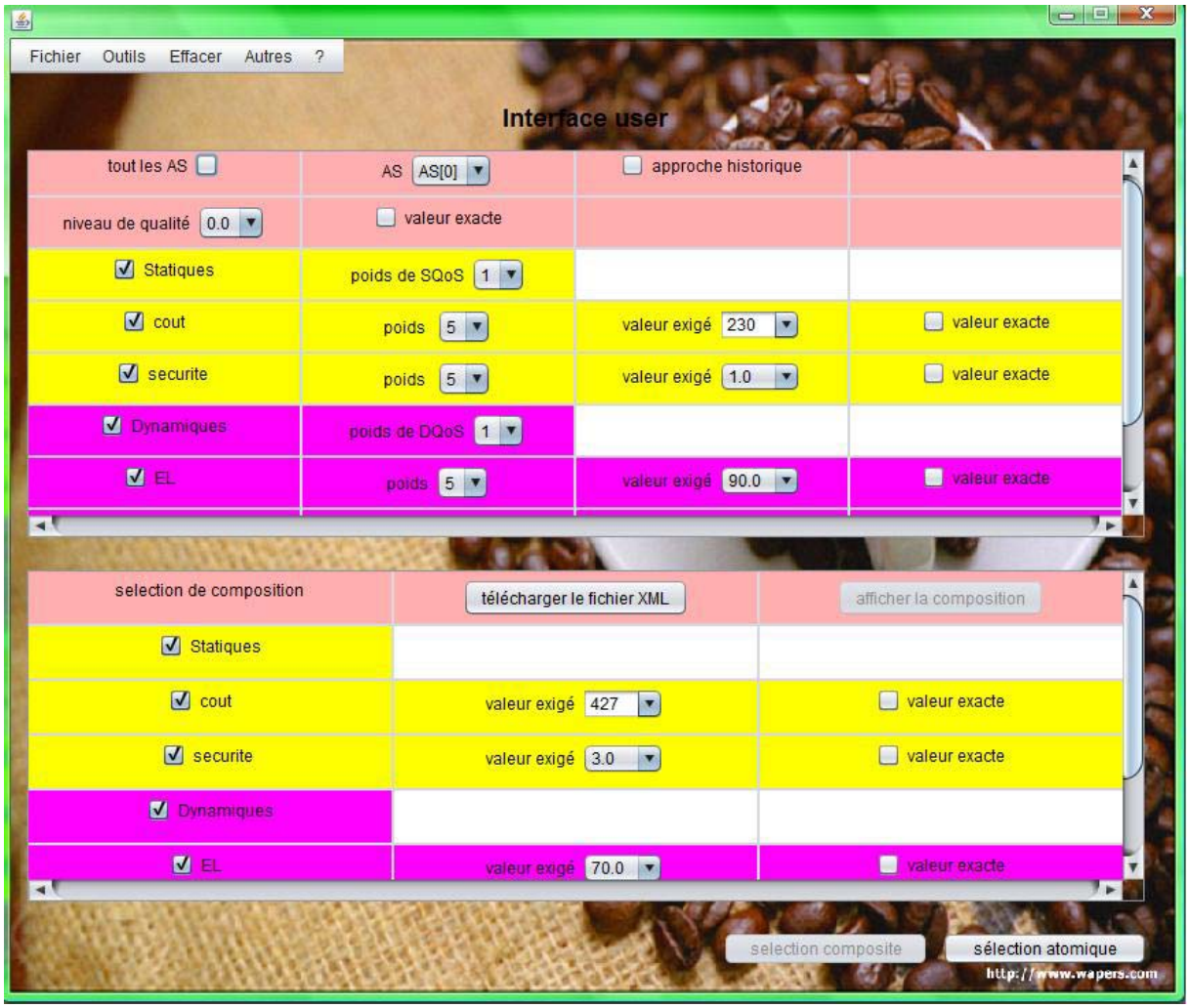


Figure IV.3. Interface utilisateur de l'application.

IV.3.1. Spécification des exigences locales

- ✓ L'utilisateur définit ses exigences locales spécifiques en choisissant un service abstrait dans le menu déroulant sinon il coche la case « tous les services abstraits » pour définir ses exigences locales générales, comme le montre la figure IV.4.

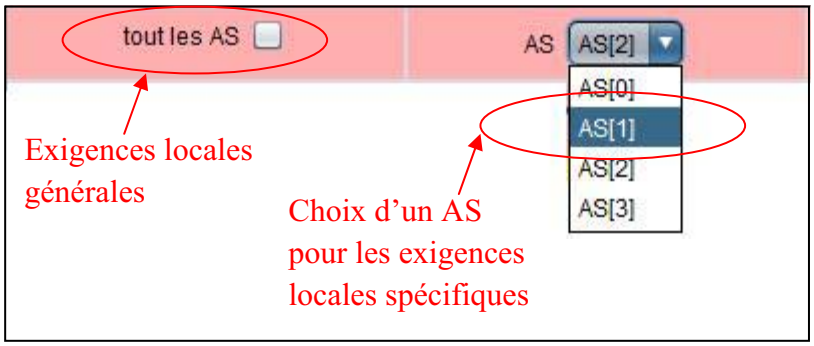


Figure IV.4. Exigences locales générales et spécifiques.

✓ L'utilisateur spécifie le niveau de qualité requis et définit la nature de son exigence (bornée valeur exacte), comme le montre la figure IV.5.

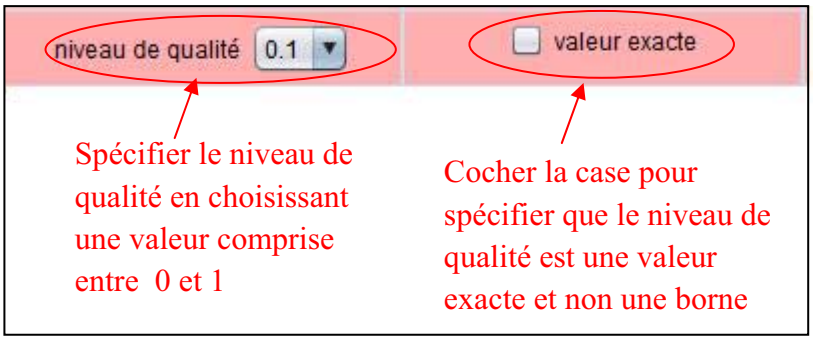


Figure IV.5. Niveau de qualité requis.

✓ L'utilisateur choisi : (Figure IV.6)

- Tout ou partie des paramètres de qualité statiques et/ou dynamiques
- Attribut un poids et définit la valeur exigée pour chaque paramètre choisi et définit la nature du paramètre (bornée valeur exacte)
- Spécifie le poids de la QoS statique et de la QoS dynamique.

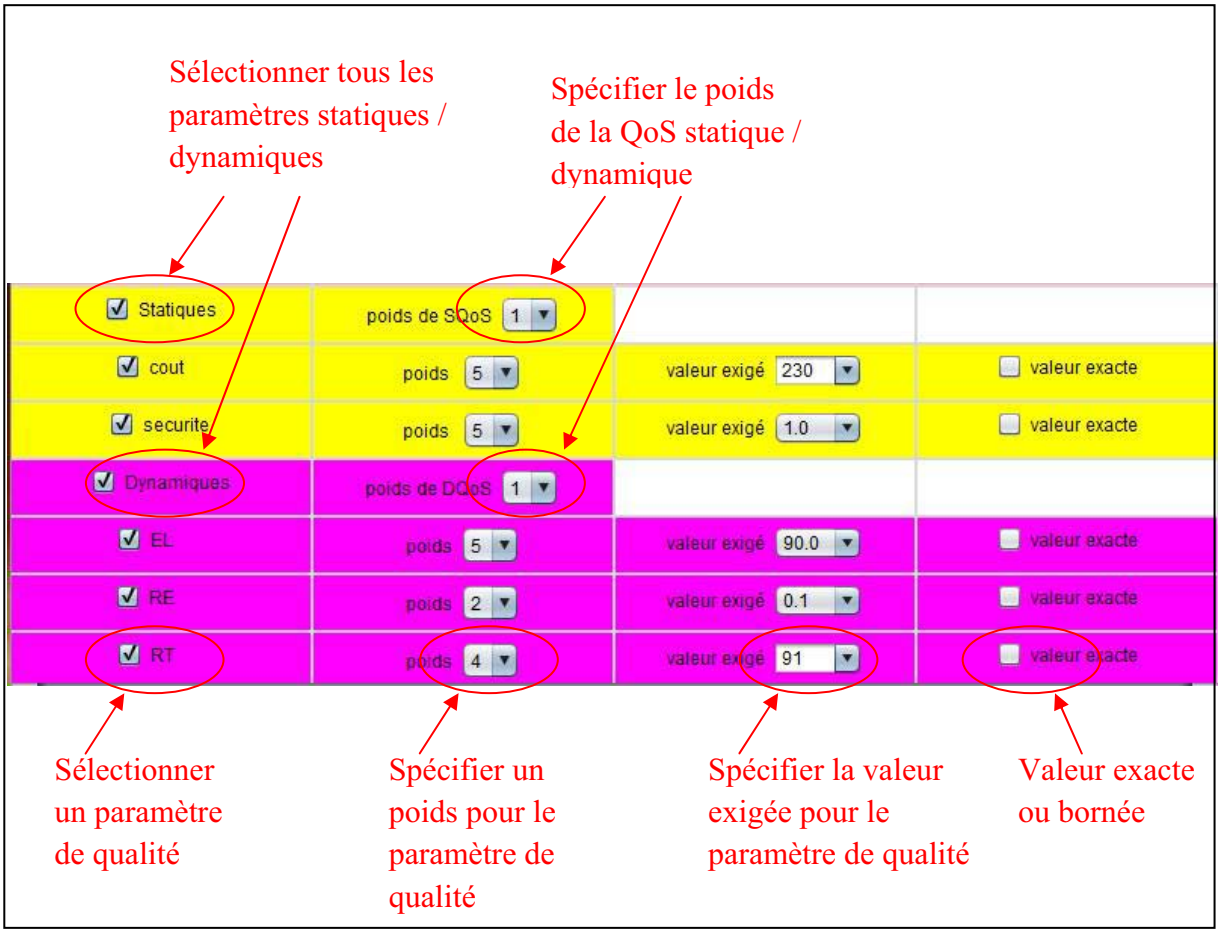


Figure IV.6. Exigences locales.

IV.3.2. Spécification des exigences globales

✓ L'utilisateur choisi : (Figure IV.7)

- Tout ou partie des paramètres de qualité statiques et/ou dynamiques
- Attribue un poids et définit la valeur exigée pour chaque paramètre choisi et définit la nature paramètre (bornée valeur exacte).

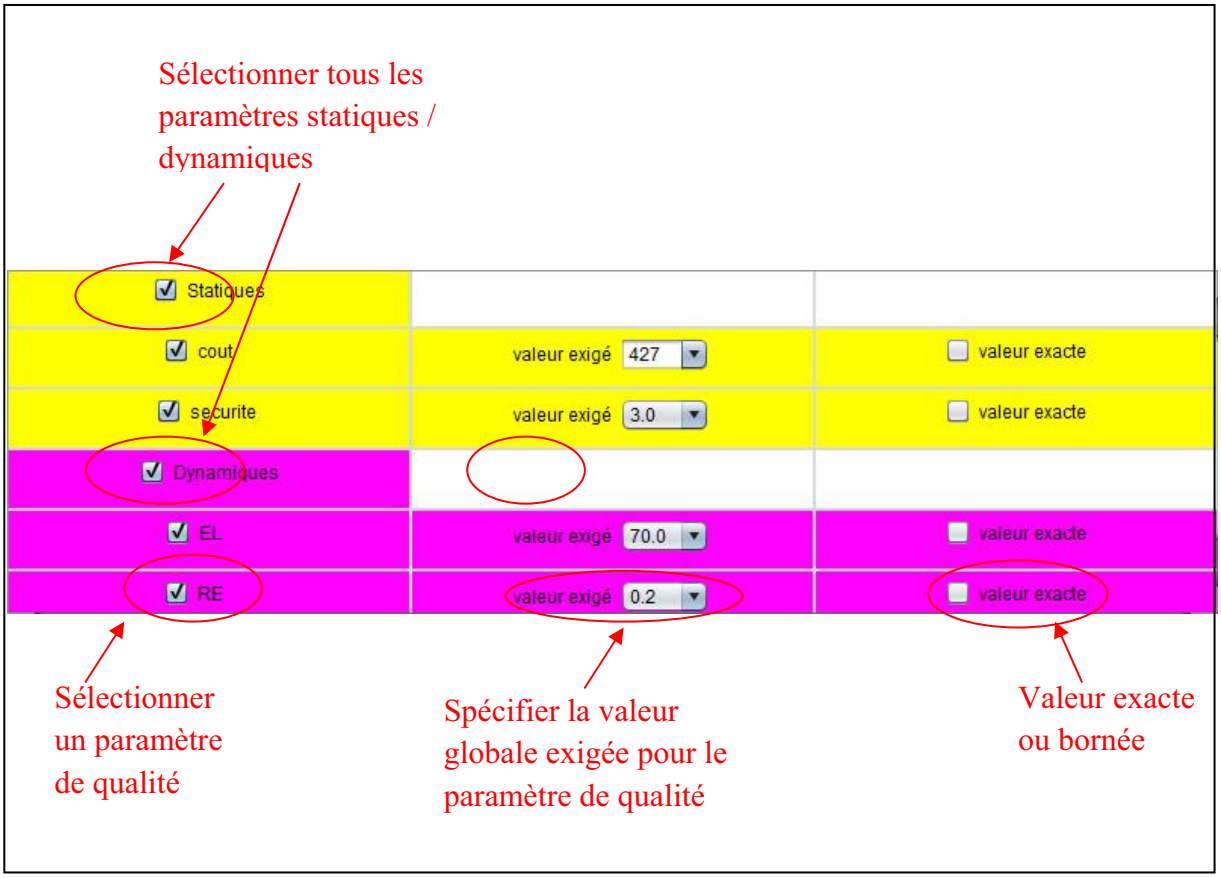


Figure IV.7. Exigences globales.

✓ L'utilisateur télécharge le fichier XML dans lequel se trouve la structure du plan abstrait, comme le montre les figures IV.8 et IV.9.

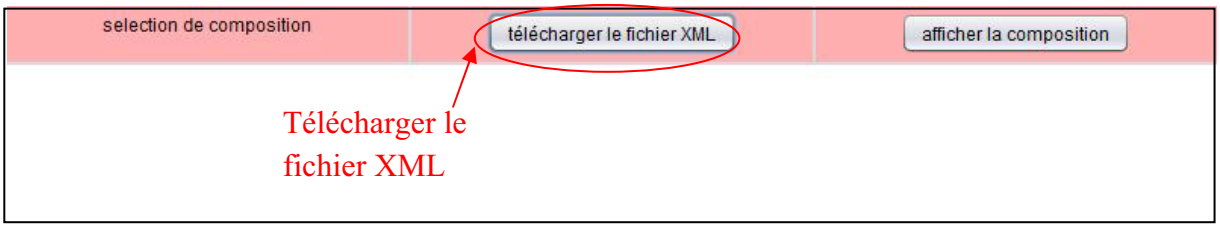


Figure IV.8. Téléchargement de fichier XML.

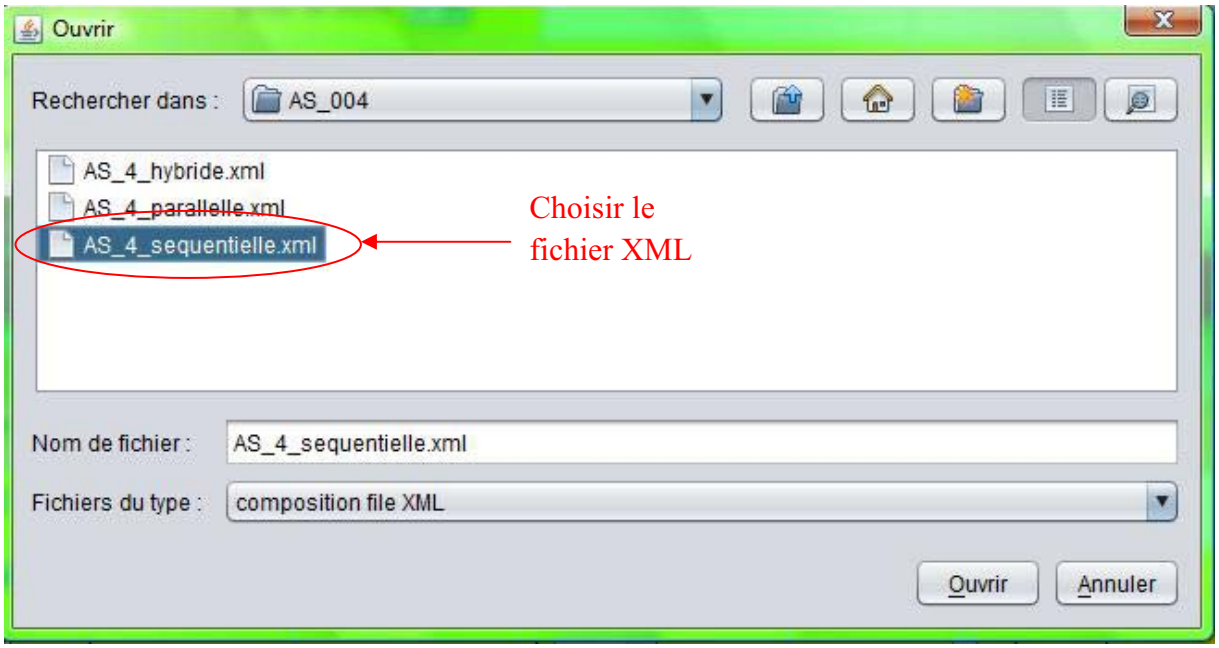


Figure IV.9. Choix du fichier XML.

✓ L'utilisateur peut afficher le plan abstrait sous la forme d'un graphe orienté, comme le montre les figures IV.10 et IV.11.

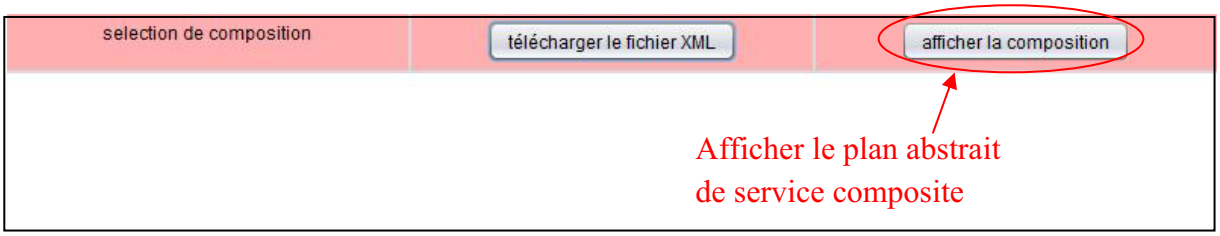


Figure IV.10. Afficher du plan abstrait.

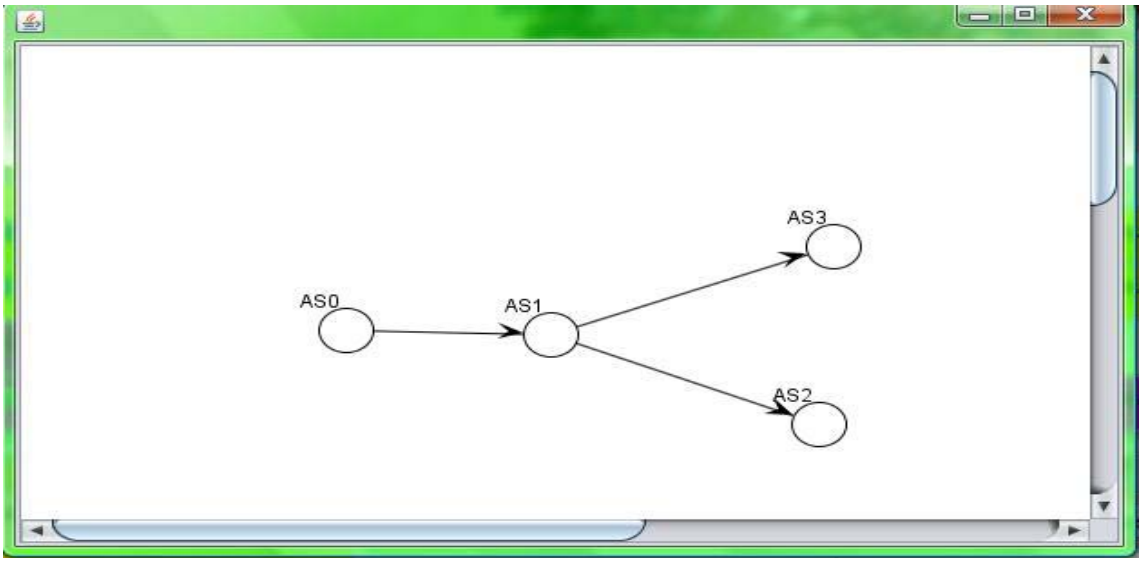


Figure IV.11. Plan abstrait d'un service composite.

IV.3.3. Lancement de la sélection de services

a. Sélection locale

✓ L'utilisateur peut effectuer une sélection locale sans vérifier les contraintes globales, comme le montre la figure IV.12.

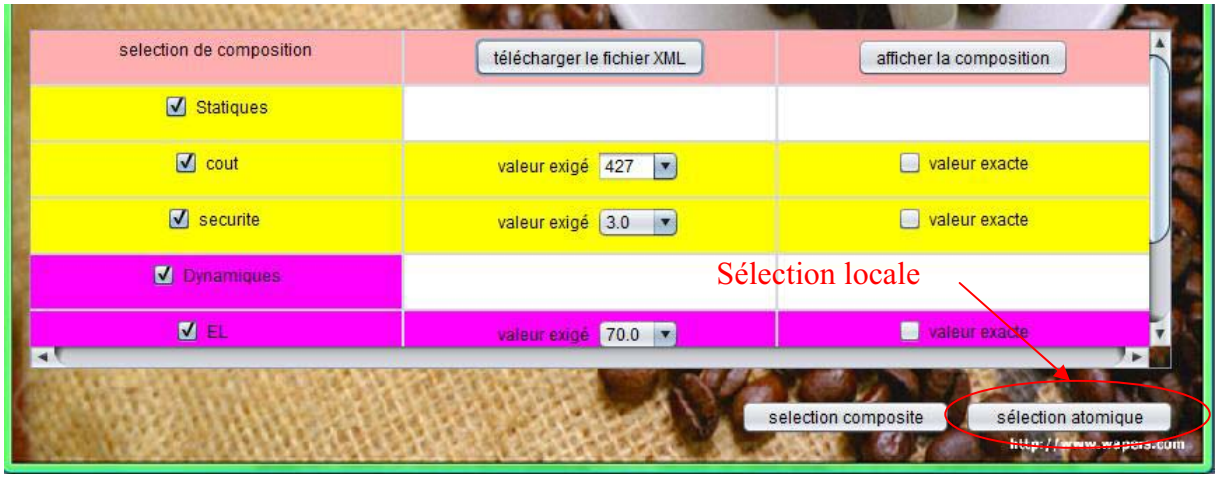


Figure IV.12. Lancement de la sélection locale.

✓ Le résultat de la sélection locale est affiché puis l'utilisateur choisi le service abstrait dans le menu et obtient : (Figure IV.13)

- L'indice du service concret sélectionné.
- Le nom du service concret sélectionné.
- La QoS statique.
- La QoS dynamique.
- La QoS.

Figure IV.13. Résultat de la sélection locale.

b. Sélection globale

✓ L'utilisateur lance la sélection globale, comme le montre la figure IV.14.

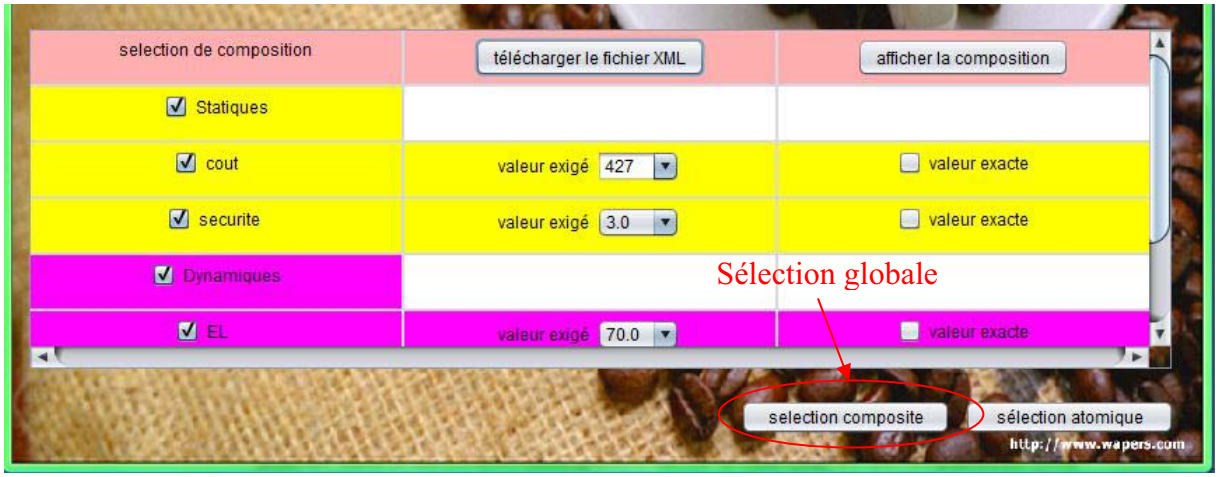


Figure IV.14. Lancement de la sélection globale.

✓ Le résultat de la sélection globale est affiché et l'utilisateur obtient :

- Le service concret sélectionné de chaque service abstrait.
- Les valeurs globales des paramètres de qualité choisis ainsi que leurs valeurs exigées.

Figure IV.15. Résultat de la sélection globale.

IV.4. Environnement de travail

L'application a été développée en JAVA 1.6 sur une machine DELL dont les caractéristiques sont les suivantes :

- Système d'exploitation : Windows 7.
- Type de système : 32 bits.
- Indice de performance Windows : 3.4.
- Processeur : Intel® core ™ 2 DUO CPU E7300 @ 2.66 GHz 2.66 GHz.
- RAM : 2.99 Go.

IV.5. Evaluation des performances

L'évaluation de notre approche consiste à calculer le temps nécessaire (en millisecondes) pour répondre aux exigences de l'utilisateur en effectuant une sélection globale.

Pour ce faire, nous avons effectué un ensemble de simulations, dans lesquelles nous avons fixé le nombre de paramètres de qualité à trois, et généré aléatoirement les différentes valeurs de paramètres de qualité.

IV.5.1. Première simulation

Cette première simulation a eu pour objectif d'évaluer notre approche par rapport aux différents modèles de plan abstrait proposés dans notre approche : séquentiel, parallèle et hybride (c.à.d. séquentiel et parallèle dans le même plan abstrait), en faisant varier le nombre de services abstraits AS de **10** à **50**, et le nombre de services concrets CS dans chaque AS de 100 à 2000.

a. Modèle séquentiel

Les résultats obtenus sont représentés dans le tableau IV.2 et la figure IV.16. Cette simulation qui concerne le modèle séquentiel montre que l'approche proposée est stable et conduit à un temps de calcul raisonnable. En effet, pour 50 services abstraits et 2000 services concrets dans chaque service abstrait, le temps de calcul est égal approximativement à 1 seconde (**1092,99** millisecondes).

b. Modèle Parallèle

Les résultats obtenus sont représentés dans le tableau IV.3 et la figure IV.17. Ils montrent que notre approche est stable et conduit à un temps de calcul raisonnable. En

considérant **50** services abstraits et **2000** services concrets dans chaque service abstrait, nous obtenons un temps de calcul qui est égal approximativement à 1 seconde (**1090,8** millisecondes).

c. Modèle Hybride

Les résultats obtenus sont illustrés sur le tableau IV.4 et la figure IV.18. Ils confirment les résultats précédents. Le temps de calcul est d'environ 1 seconde (**1094,71** millisecondes) pour une configuration avec 50 services abstraits et 2000 services concrets dans chaque service abstrait.

Séquentiel			
Nombre de CS	**Temps en ms pour 10 AS**	**Temps en ms pour 20 AS**	**Temps en ms pour 50 AS**
100	5,36	10,82	17,03
200	13,38	17,98	31,17
300	19,16	27,14	51,16
400	24,12	36,53	73,86
500	29,77	47,53	100,06
600	35,96	60,77	131,63
700	44,64	75,52	169,38
800	52,92	92,63	211,57
900	62,44	110,64	258,03
1000	81,68	133,27	308,36
1100	85,02	157,5	373
1200	98,63	181,96	428,22
1300	112,17	215,41	456,49
1400	137,7	241,04	566,24
1500	141,93	265,98	639,87
1600	158,52	308,88	725,92
1700	180,76	347,47	813,49
1800	196,55	371,28	936,39
1900	214,37	413,03	1011,88
2000	235,86	490,42	1092,99

Tableau IV.2. Modèle séquentiel pour la simulation 1.

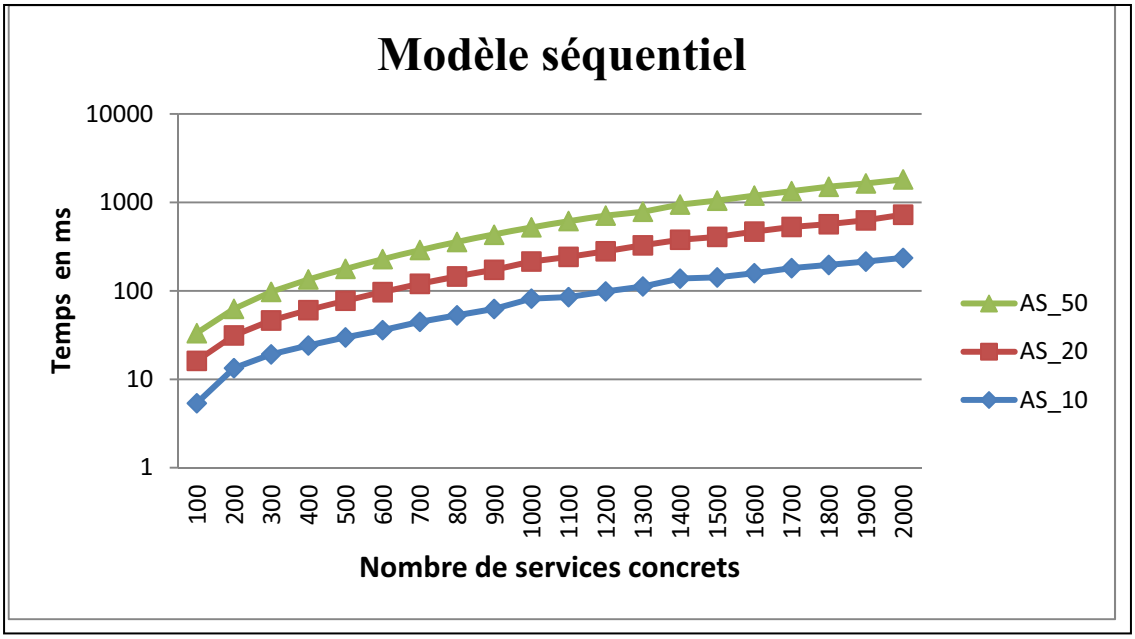


Figure IV.16. Performances dans le cas du modèle séquentiel pour la simulation 1.

Parallèle			
Nombre de CS	**Temps en ms pour 10 AS**	**Temps en ms pour 20 AS**	**Temps en ms pour 50 AS**
100	6,67	9,82	17,13
200	13,58	17,48	31,71
300	19,06	27,41	52,06
400	23,93	37,53	73,86
500	29,69	46,53	98,16
600	36,39	61,77	132,63
700	52,54	73,52	171,38
800	52,51	90,63	213,57
900	62,54	112,64	257,07
1000	75,28	135,87	309,86
1100	85,57	156,05	374,76
1200	99,61	181,96	428,92
1300	111,94	214,41	456,14
1400	127,78	240,14	566,74
1500	142,13	263,98	638,11
1600	160,17	306,88	724,91
1700	176,86	345,47	814,08
1800	195,9	373,18	935,29
1900	213,49	410,93	1012,76
2000	234,44	491,47	1090,8

Tableau IV.3. Modèle parallèle pour la simulation 1.

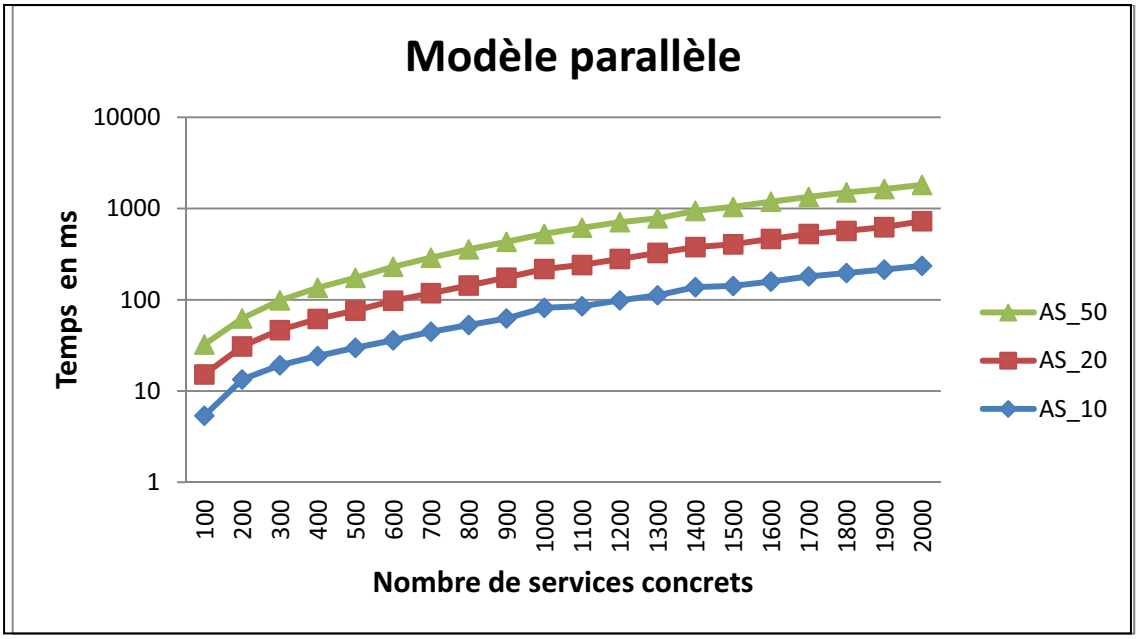


Figure IV.17. Performances dans le cas du modèle parallèle pour la simulation 1.

Hybride			
Nombre de CS	**Temps en ms pour 10 AS**	**Temps en ms pour 20 AS**	**Temps en ms pour 50 AS**
100	9,94	16,38	24,56
200	13,19	18,2	31,61
300	27,68	29,19	50,92
400	29,93	37,53	73,84
500	31,8	53,72	100,53
600	36,39	65,77	135,68
700	50,54	73,52	169,46
800	52,51	90,63	200,5
900	62,54	112,64	268,12
1000	81,35	137,08	308,22
1100	90,57	157,05	404,54
1200	99,69	182,96	428,27
1300	113,94	217,41	429,23
1400	129,78	242,14	583,58
1500	143,13	265,98	657,61
1600	163,17	309,88	769,31
1700	179,86	347,47	803,3
1800	195,19	373,18	899,07
1900	215,49	406,93	1000,83
2000	235,81	452,2	1094,71

Tableau IV.4. Modèle hybride pour la simulation 1.

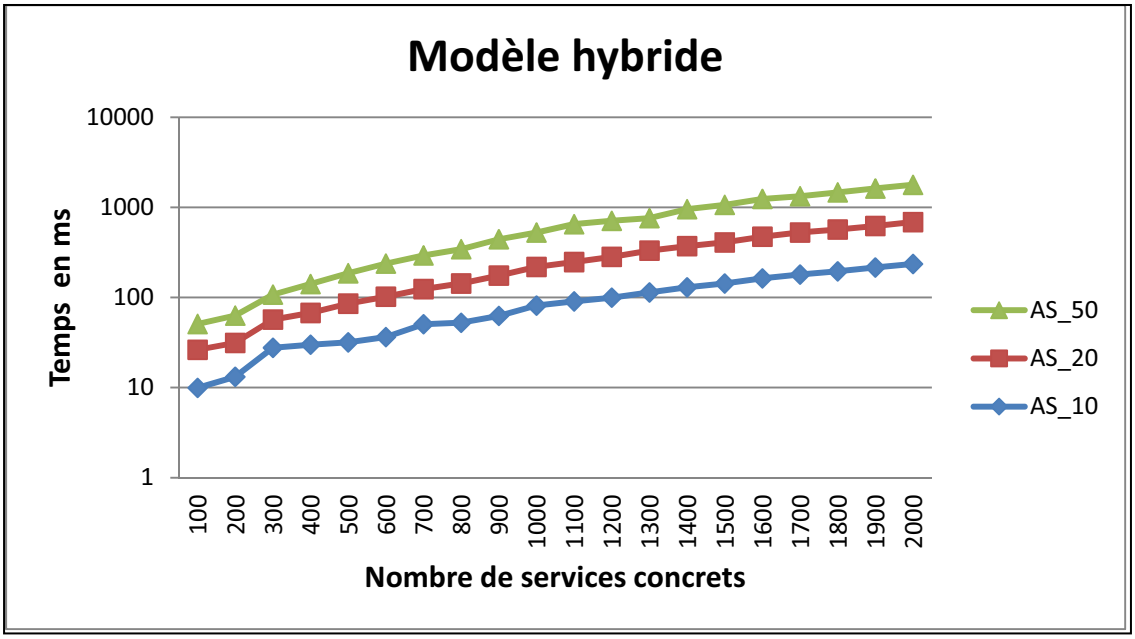


Figure IV.18. Performances dans le cas du modèle hybride pour la simulation 1.

IV.5.2. Deuxième simulation

Dans cette deuxième simulation, nous essayons de montrer l'efficacité de l'approche de sélection en termes de temps de réponse en considérant un nombre très élevé de services concrets. L'approche est évaluée sur différents modèles d'un plan abstrait : séquentiel, parallèle et hybride. Dans cette simulation, le nombre des services abstraits AS varie de **5** à **50**, et le nombre de services concrets CS dans chaque AS de **10** à **5000**.

a. Modèle séquentiel

Les résultats obtenus relatifs au modèle séquentiel sont présentés dans le tableau IV.5 et la figure IV.19. Malgré l'augmentation notable du temps d'exécution, après 2000 service concrets, les résultats montrent que l'approche proposée garantit un temps de calcul raisonnable. En effet, pour **50** services abstraits et **5000** services concrets dans chaque service abstrait, le temps de calcul est approximativement égal à 6 secondes (**6573,97** millisecondes).

Séquentiel Nbre de AS / Nbre de CS	5	10	20	30	40	50
10	0,85	1,76	2,98	8,87	22,12	30,49
50	1,57	2,32	3,42	13,71	28,93	30,47
100	3,07	6,54	22,06	8,91	20,42	55,66
500	40,05	83,02	164,88	250,39	88,44	299,42
1000	149,6	299,29	128,06	236,95	256,58	325,67
2000	133,55	245,15	463,92	759,87	867,05	1087,44
3000	233,57	472,54	959,01	1450,52	1882,62	2714,99
4000	450,86	822,93	1937,68	2785,8	3444,39	4234,29
5000	642,37	1303,24	2683,55	4397,33	6425,16	6573,97

Tableau IV.5. Modèle séquentiel pour la simulation 2.

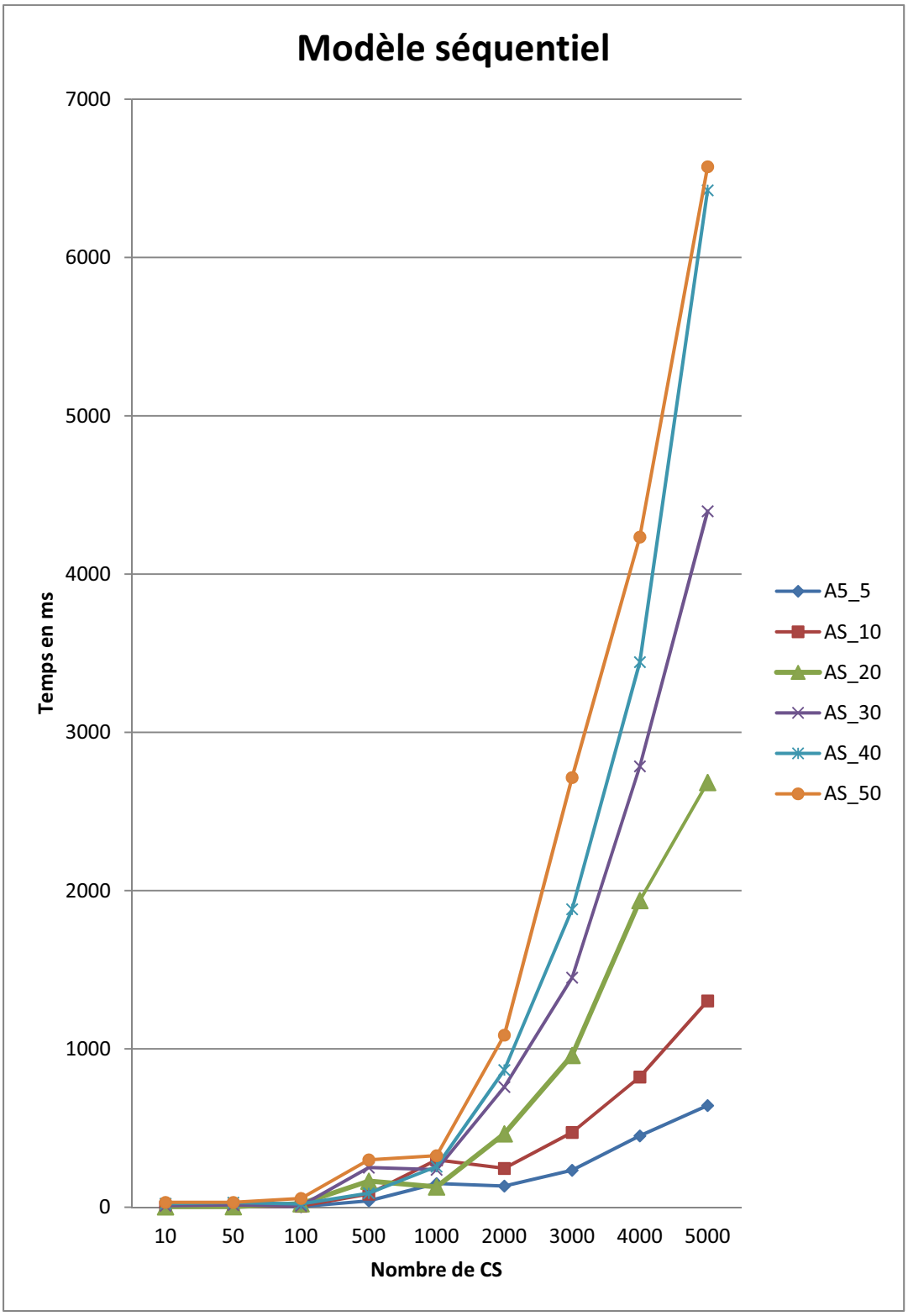


Figure IV.19. Performances dans le cas du modèle séquentiel pour la simulation 2.

b. Modèle Parallèle

Les résultats concernant le modèle parallèle sont représentés dans le tableau IV.6 et la figure IV.20. Les performances sont sensiblement les mêmes que pour le modèle séquentiel.

Parallèle Nbre de AS / Nbre de CS	5	10	20	30	40	50
10	0,8	1,92	4,34	19,26	14,63	23,21
50	1,75	2,89	5,3	14,28	14,86	24,13
100	3,17	7,79	14,79	8,81	18,05	54,75
500	40,11	18,04	171,76	250,8	78,91	310,39
1000	153,13	300,23	129,74	238,37	272,79	311,31
2000	106,78	220,08	433,24	717,48	873,48	1072,12
3000	242,5	470,79	957,73	1517,69	1925,1	2411,17
4000	411,25	949,08	1712,31	2807,83	3441,54	4170,44
5000	654,73	1303,2	2628,08	4373,56	6423,23	6444,12

Tableau IV.6. Modèle séquentiel pour la simulation 2.

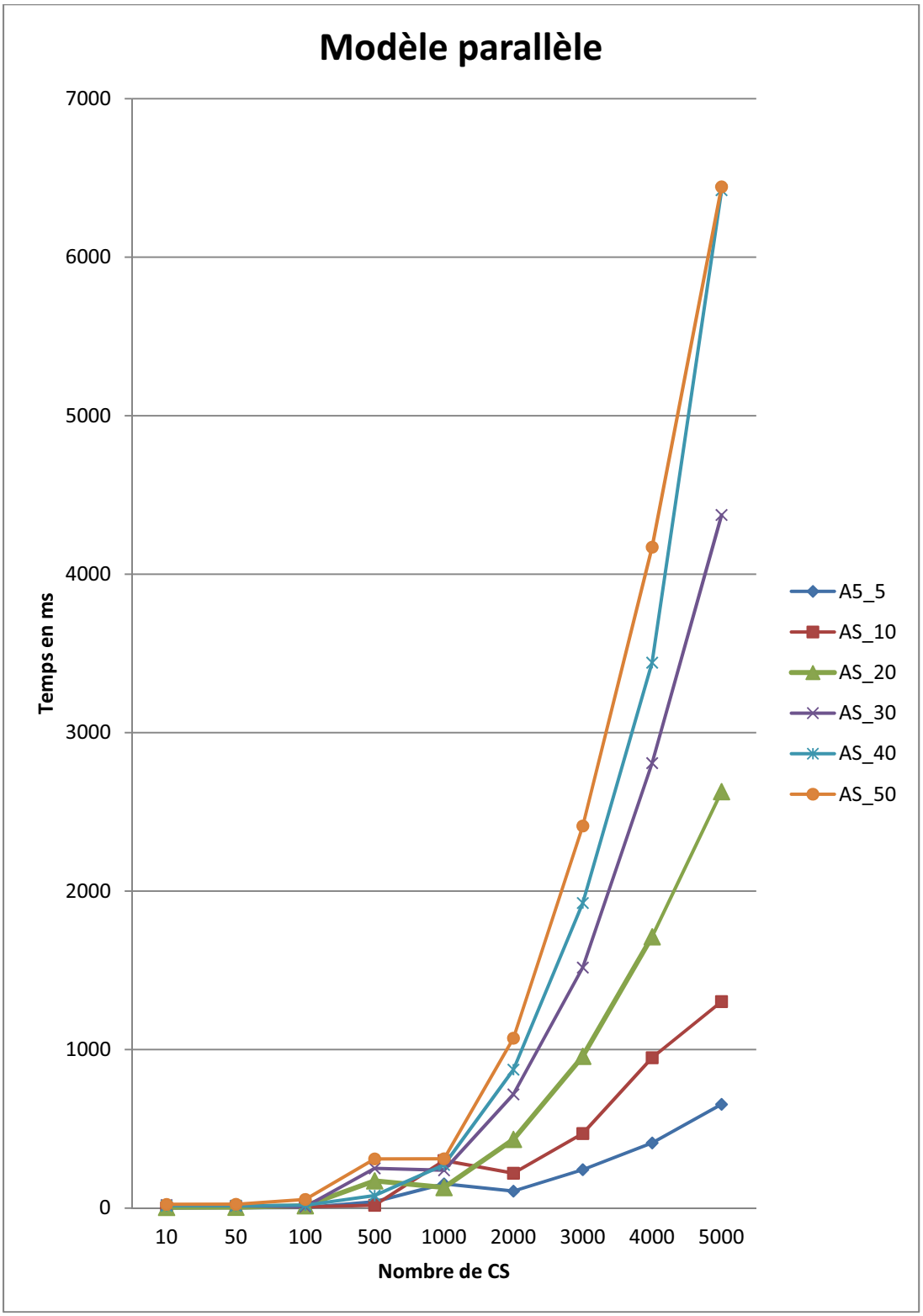


Figure IV.20. Performances dans le cas du modèle parallèle pour la simulation 2.

c. Modèle Hybride

Les résultats obtenus dans le cas du modèle hybride sont illustrés dans le tableau IV.7 et la figure IV.21. Ces résultats qui sont sensiblement proches de ceux obtenus dans le cas des modèles parallèle et séquentiel, confirment les bonnes performances de l'approche proposée.

Hybride Nbre de AS / Nbre de CS	5	10	20	30	40	50
10	1,01	2,42	7,08	12,04	26,44	35,01
50	1,92	2,72	11,8	24,3	15,67	40,18
100	3,42	9,09	16,38	28,45	20,72	65,92
500	40,14	32,31	52,8	64,72	80,52	307,9
1000	153,25	81,35	137,08	215,36	253,19	311,72
2000	127,08	245,86	456,34	700,03	908,93	1088,59
3000	265,32	515,44	997,44	1454,52	1885,86	2451,55
4000	449,92	859,51	1739,74	2775,36	3609,98	4125,46
5000	743,71	1325,66	2950,33	4226,1	6435,65	6491,06

Tableau IV.7. Modèle séquentiel pour la simulation 2.

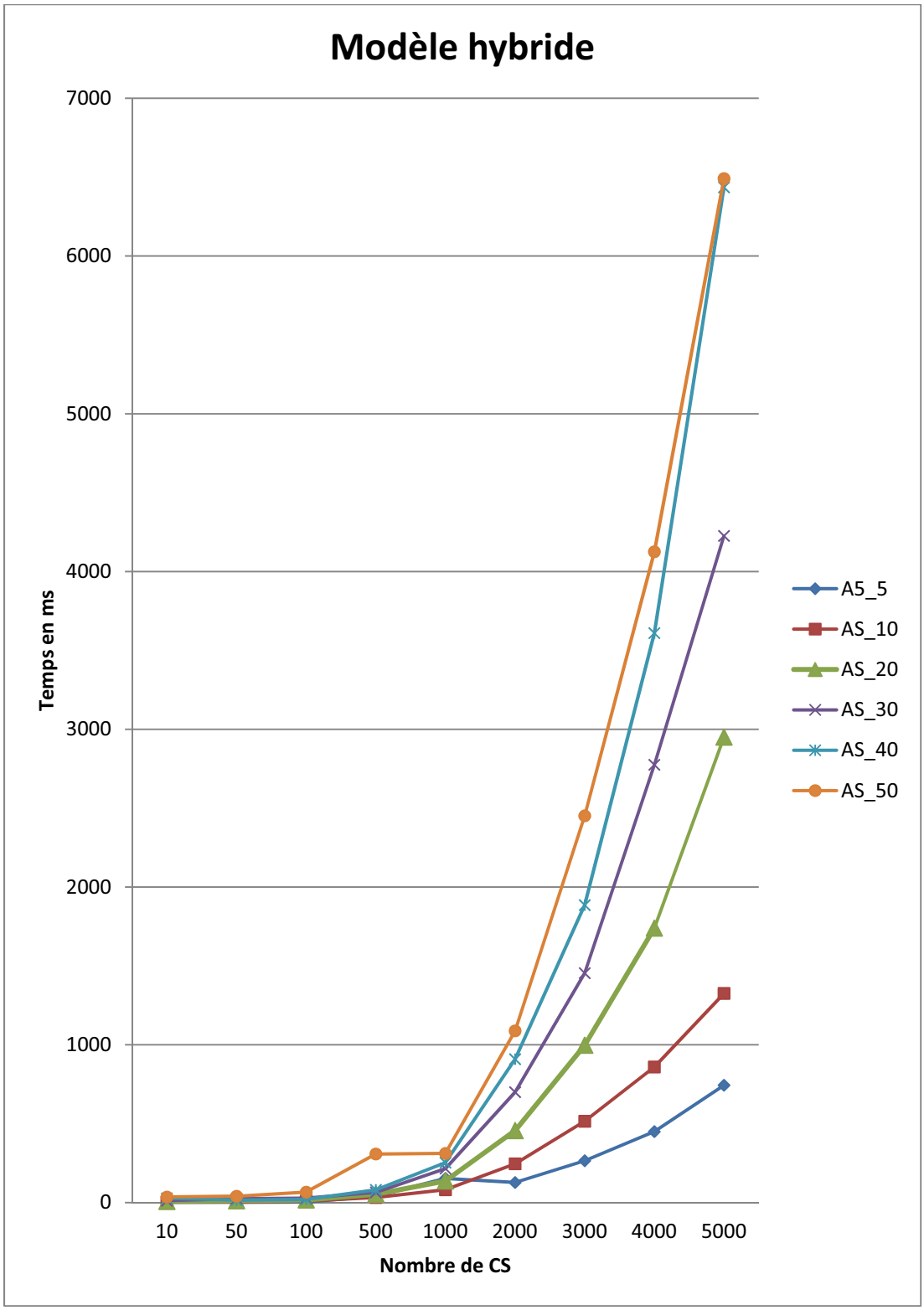


Figure IV.21. Performances dans le cas du modèle hybride pour la simulation 2.

IV.6. Comparaison

Pour montrer l'efficacité de l'approche proposée, nous précédons à une comparaison avec les résultats obtenus dans **[Zeng, 2004], [Ardagna, 2007].** Pour ce faire, nous nous plaçons dans les mêmes conditions de simulations. Cette étude comparative est effectuée sur un modèle séquentiel dont le nombre de services abstraits est fixé à 10 et le nombre de services concrets varie entre 100 et 2000.

Les résultats obtenus sont représentés dans le tableau IV.8 et la figure IV.22. Ils montrent la supériorité de l'approche proposée par rapport à celles de **[Zeng, 2004], [Ardagna, 2007]**. Nous observons une nette amélioration du temps de calcul. Ce dernier est, en effet, réduit d'un facteur 300 par rapport aux approches de **[Zeng, 2004], [Ardagna, 2007].**

Séquentiel	Notre approche	Les approches de [Zeng, 2004], [Ardagna, 2007]
Nombre de CS	Temps en ms pour 10 AS	Temps en ms pour 10 AS
100	5,36	100
200	13,38	600
300	19,16	1000
400	24,12	5000
500	29,77	4200
600	35,96	6000
700	44,64	6800
800	52,92	9000
900	62,44	9500
1000	81,68	10000
1100	85,02	20000
1200	98,63	30000
1300	112,17	35000
1400	137,7	45000
1500	141,93	50000
1600	158,52	56000
1700	180,76	60000
1800	196,55	62000
1900	214,37	68000
2000	235,86	61000

Tableau IV.8. Comparaison des performances.

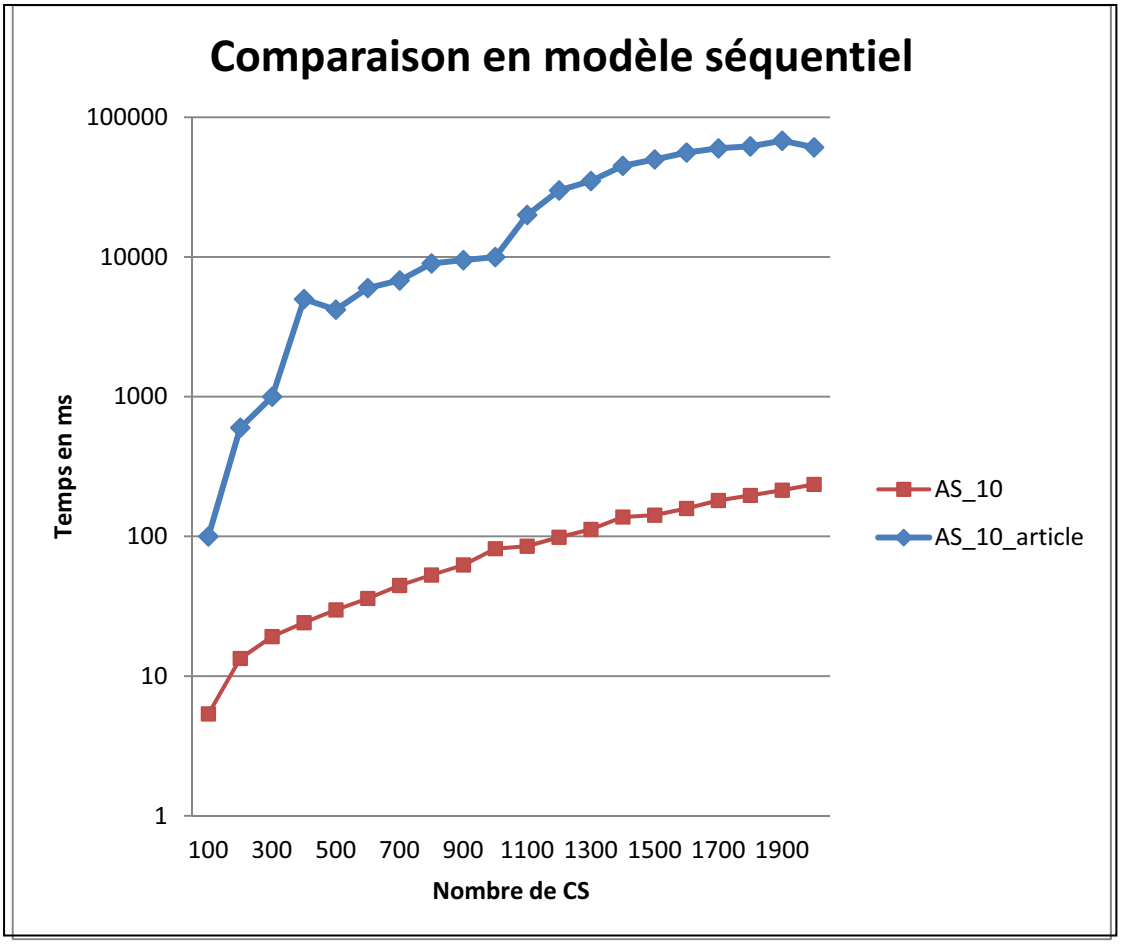


Figure IV.22. Comparaison des performances dans le cas du modèle séquentiel.

IV.7. Conclusion

Dans ce chapitre, nous avons présenté l'application ayant servi de support pour l'évaluation des performances de l'approche proposée. A travers les simulations effectuées, nous avons montré l'efficacité de notre approche et sa supériorité par rapport aux approches de **[Zeng, 2004], [Ardagna, 2007]**.

Conclusion générale

Dans ce mémoire, nous avons proposé une approche de sélection globale de services, basée sur le formalisme CSP. Cette approche permet d'une part, d'effectuer une sélection locale en prenant en considération les préférences et les exigences locales de l'utilisateur et d'autre part, de vérifier les exigences globales et de choisir le service composite optimal en termes de qualité de service.

Cette approche prend en considération différents modèles du plan abstrait : séquentiel, parallèle, boucle et hybride. Elle prend également en compte plusieurs paramètres de qualité positifs et négatifs.

Cette approche permet aussi une sélection dynamique des services en prenant en considération les pannes qui peuvent se produire lors de l'invocation des différents services sélectionnés.

La sélection, utilise par ailleurs les informations de contexte, comme le niveau d'énergie des équipements fournissant les services.

Bien que cette approche soit basée sur un algorithme dont la complexité est exponentielle (problème multi critères), les résultats obtenus en termes de temps de calcul restent satisfaisants grâce aux heuristiques utilisés.

Le passage à l'échelle de l'approche ne peut être considéré comme un problème étant donné le petit nombre de services traités dans les environnements ubiquitaires.

Une perspective à ce travail consiste à proposer une amélioration de l'approche pour supporter le passage à l'échelle, dans le but de l'utiliser dans des environnements à très grand nombre de services tels que le cloud computing.

Bibliographie

[Want, 2005]: R.Want, T. Pering. *System Challenges for Ubiquitous & Pervasing Computing*. Proceedings of the 27th international conference on Software engineering, (ICSE'05), pp 9-14, NEW York, NY, USA. ACM Press, 2005.

[Saha, 2003]: D. Saha, A. Mukherjee. *Pervasive Computing: A Paradigm for the 21st Century*. IEEE Computer Society, 36(2):25-31, 2003.

[Papazoglou, 2003] : *M. P. Papazoglou, D. Georgakopoulos.* Service-Oriented Computing. Communications of the ACM, 46(10), 2003.

[Satyanarayanan, 2001] : M. Satyanarayanan. *Ubiquitous Computing : les challenges logiciels*. *IEEE Personnal communication*, p. 10-17, August 2001.

[Want, 1992b]: R. Want, A. Hopper, V. Falcão et J. Gibbons. 1992b. *The Active Badge Location System*. *ACM Transaction on Information Systems,* vol. 10, p. 91-102, Octobre 1992.

[Dey, 2000]: Dey, A. K., et G. D. Abowd. *Towards a Better Understanding of Context and Context-Awareness*. In Workshop on the What, Who, Where, When, and How of Context Awareness Conference on Human Factors in Computer Systems (CHI2000), The Hague, Netherlands, 2000.

[Sadeh, 2002]: N. Sadeh., E. Chan, Y. Shimazaki et al. *Mycampus: An agent-based environment for context-aware mobile services*. In: the proceedings of the AAMAS02 Workshop on Ubiquitous Agents on Embedded, Wearable, and Mobile Devices, Bologna, Italy, July 2002.

[Pascoe, 1998] : J. Pascoe. *Adding Generic Contextual Capabilities to Wearable Computers*. In *Symposium on Wearable Computers (ISWC'98)* (October 1998). p. 92 - 99. Pittsburgh, Pennsylvania USA, 1998.

[Miraoui, 2009] : M. Miraoui. *Architecture logicielle pour l'informatique diffuse : modélisation du contexte et adaptation dynamique des services*. Thèse de doctorat en informatique, école de technologie supérieure université du Québec, Montréal, juillet 2009.

[Yérom-David, 2006]: B. Yérom-David. *Résolution de l'hétérogénéité des intergiciels d'un environnement ubiquitaire*. Thèse de doctorat en informatique, l'université de Versailles Saint-Quentin-en-Yvelines, France, 2006.

[IBM , 2006] : IBM and CHU in Nice, http://www.ibm.com/news/fr/fr/2006/03/cp1851.html. New Generation Hospital project, 2006.

[Costa, 2007]: P. Costa, G. Coulson, R. Gold, M. Lad, C. Mascolo, L. Mottola, GP. Picco, T. Sivaharan, N. Weerasinghe and S. Zachariadis. *The RUNES Middleware for Networked Embedded Systems and its Application in a Disaster Management Scenario*. In Proceedings of 5th Annual IEEE International Conference on Pervasive Computing and Communications, pages 69-78, 2007.

[Chollet, 2009] : S. Chollet : *Orchestration de Services Hétérogènes et Sécurisés*, Pages 20-21, Thèse en Informatique, Equipe ADELE du LIG - Université Joseph Fourier, Grenoble, France*,* Décembre. 2009.

[Redmond, 2002] : B. Redmond and V. Cahill, *Supporting Unanticipated Dynamic Adaptation of Application Behaviour, ECOOP, Lecture Notes in Computer Science,Vol. 2374, pp. 205-230, Springer,* 2002.

[Mrissa, 2007] : M. Mrissa, *Médiation Sémantique Orientée Contexte pour la Composition de Services Web.* Thèse doctorat**,** Université Claude Bernard Lyon I.2007.

[Charif, 2007] : Y.Charif, *Chorégraphie dynamique de services basée sur la coordination d'agents introspectifs.* Thèse doctorat*,* université pierre et marie curie paris vi.2007

[Zeng, 2004]: L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. *Qos-aware middleware for web services composition*. IEEE Transactions on Software Engineering, 30(5):311–327, 2004.

[Ardagna, 2007]: D. Ardagna and B. Pernici. *Adaptive service composition in flexible processes*. IEEE Transactions on Software Engineering, 33(6):369–384, 2007.

[Maros, 2003]: I. Maros. *Computational Techniques of the Simplex Method*. Springer, 2003.

[Wang, 2006]: X. Wang, T. Vitvar, M. Kerrigan, and I. Toma, *A QoS-Aware Selection Model for Semantic Web Services, ICSOC, Lecture Notes in Computer Science, Vol. 4294, pp. 390-401,Springer,* 2006.

[Eyhab, 2007]: A. Eyhab and Qusay H. Mahmoud. *Discovering the best web service*. *InWWW '07: Proceedings of the 16th international conference on World Wide Web,pages 1257–1258. ACM*, 2007.

[Wang, 2007]: Y. Wang, J. Vassileva ,*Toward Trust and Reputation Based Web Service Selection*: A Survey*, Proc International Transactions on Systems Science and Applications ITSSA Journal***,** 2007.

[Rosenberg, 2009]: F. Rosenberg, P. Leitner, A. Michlmayr, P. Celikovic and S. Dustdar. *Towards Composition as a Service – A Quality of Service Driven Approach*. Distributed Systems Group, Technical University Vienna Argentinierstrasse 8/184-1, 1040 Vienna, Austria, 2009.

[Zhang, 2007]: T. Yu, Y. Zhang, and K.-J. Lin. Efficient algorithms for web services selection with end-to-end qos constraints. ACM Transactions on the Web, 1(1), 2007.

[B. Mabrouk, 2009]: N. B. Mabrouk, S. Beauche, E. Kuznetsova, N. Georgantas, and V. Issarny, *QoS-Aware Service Composition in Dynamic Service Oriented Environments*, *Middleware, Lecture Notes in Computer Science, Vol. 5896, pp. 123-142, Springer 4*, novembre 2009.

[Michael, 2007]: C. J. Michael and G. Muhl*, QoS-based Selection of Services:The Implementation of a Genetic Algorithm, KiVS 2007 Workshop: Service-Oriente Architectures und ServiceOriented Computing (SOA/SOC)* 2007.

[Michael, 2004]: C. J. Michael, G Rojec-Goldmann, and G. Muhl, *QoS Aggregation for Web Service Composition using Workflow Patterns, EDOC, pp. 149-159, IEEE Computer Society,*2004.

[B. Mokhtar**, 2007]**: S. B. Mokhtar, N. Georgantas, V. Issarny, *COCOA: COnversation-based service Composition in pervAsive computing environments with QoS support*, ScienceDirect, The Journal of Systems and Software, 2007.

[Wang, 2006] : Y. Wang, H. Wang, X. Xu ,*Web Services Selection and Composition based on the Routing Algorithm, EDOC Workshops, p. 69, IEEE Computer Society,* 2006.

[Alrifai, 2009]: M. Alrifai and T. Riss, *Combining Global Optimization with Local Selection for Efficient QoS-aware Service Composition* , WWW 2009, *Madrid*, Spain, 2009.