République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université A/Mira de Béjaïa
Faculté des Sciences Exactes
Département d'Informatique

Mémoire de fin de cycle
en vue d'obtention du diplôme de master recherche en informatique
spécialité : Réseaux et systèmes distribués

Thème

Composition de services avec QoS globale basée sur les algorithmes génétiques dans les environnements ubiquitaires.

Mémoire soutenu le 24/06/2014 par :

M^r ***BENAYACHE*** **Abbes**

M^{elle} ***SLIMANI*** **Sarah**

Devant le jury composé de :

Président	M^r ***MIR*** **Foudil**	**M.A.A, U.A.M Béjaia**
Examinateur	M^r ***ABBACHE*** **Bournane**	**M.A.A, U.A.M Béjaia**
Examinateur	M^r ***FARAH*** **Zoubyer**	**M.A.A, U.A.M Béjaia**
Rapporteur	M^r ***KHANOUCHE*** **M-Essaid**	**M.A.A, U.A.M Béjaia**

Promotion 2013/2014

Résumé

Les systèmes ubiquitaires ont pour objectif de rendre l'information disponible partout et à tout moment, et d'offrir une multitude de services afin d'améliorer la qualité de vie et le bien être des utilisateurs. La prolifération du nombre des services fournissant les mêmes fonctionnalités mais une qualité de service (QoS) différentes, rend complexe la tâche de sélection et de composition de service. En effet, lorsqu'un utilisateur requiert un service qui n'est pas directement disponible dans son environnement, le processus de composition sélectionne des services particuliers parmi un ensemble de services et les combine afin de former un service composite répondant au besoin complexe de l'utilisateur. Dans ce travail, nous proposons QSCGA (pour QoS-aware Service Composition based Genetic Algorithm), une approche de sélection pour la composition automatique et dynamique de services. Elle traite les exigences locales imposées par l'utilisateur en définissant un seuil de qualité de service. Au niveau global, la solution se base sur l'utilisation du principe des algorithmes génétiques (sélection, mutation, croisement et remplacement) pour avoir une solution (service composite) maximisant les valeurs de QoS. Les scénarios de simulation ont montrés les performances élevées de l'algorithme proposé. Les résultats obtenus ont été comparés avec l'une des approches proposées dans la littérature, ils montrent une amélioration en termes de temps de sélection.

Mots-clés : Environnement ubiquitaire, Qualité de service, Composition et sélection de services, Architecture orientée services, Sensibilité au contexte, Algorithme génétique.

Abtract

Ubiquitous environments aim to make information available anywhere and at any time, and offer a variety of services to improve the users life quality and well-being. The proliferation of many services providing the same functionality but a different quality of service (QoS), making complexe the selection and service composition tasks. Indeed, when a user requests a service that is not directly available in the environment, the composition process selects specific services from a set of services and combines them to form a composite service that meets the complex needs of user. In this work, we propose QSCGA (QoS-aware service composition based Genetic Algorithm), a selection approach for automatic and dynamic service composition. It addresses the local requirements of the user in defining a QoS threshold. At the global level, the solution is based on the use of the genetic algorithms principle (selection, mutation, crossover and replacement) to have a solution (composite service) maximizing the QoS values. The validation scenarios have shown the high performance of the proposed algorithm. The results obtained were compared with one of the approaches proposed in the literature, they show an improvement in terms of the selection time.

Keywords : Ubiquitous environment, Quality of service, Service composition and selection, Service oriented architecture, Context-aware, Genetic algorithm.

Remerciement

"Je m'efforce de tout comprendre
et de ne rien condamner.

Marcel Proust

Nous tenons dans un premier temps à remercier le Dieu tout puissant qui nous a donné le courage et la volonté pour mener à bien ce modeste travail.

Ce mémoire n'aurait jamais pu voir le jour sans le soutien actif d'un certain nombre de personnes que nous tenons à remercier, toutes celles et ceux qui ont contribué à la réalisation de ce modest travail :

Nos chers parents qui nous ont encouragé et supporté durant toute cette période.

Notre promoteur, en l'occurrence Mr. KHANOUCHE M^{ed} Essaid qui nous a inculqué une grande confiance et nous a orienté dans le bon sens quant à l'élaboration de ce projet.

Les membres de jury qui ont accepté d'évaluer notre travail.

Tout le personnels de départements Informatique, en particulier nos enseignants qui se sont tellement donnés durant ces 5 ans de formation pour nous transmettre se riche savoir.

Nous remerçions ainsi tous nos amis qui nous ont encouragé et aidé, en particulier Sarah Oulhaci.

Dédicaces

Je dédie ce modeste travail :

***A** la mémoire de mon très chèr père qui a toujours souhaité voir ce jour,*

***A** ma très chère maman qui m'a toujours soutenu et encouragé, et sans elle je ne serai jamais ce que je suis aujourd'hui,*

***A** mes frères adorés, leurs femmes et leurs enfants,*

***A** mon petit frères HAMZA qui a toujours été là pour moi,*

***A** ma chère et unique soeur MERIEM, son mari et ses filles,*

***A** toutes ma famille et **A** tous mes amis(es),*

***A** mon binôme Abbes,*

Enfin, ***A** toutes les personnes qui nous ont apporté de l'aide.*

Sarah

Je dédie ce modeste travail :

***A** mes chers parents,*

***A** mes frères et à mes sœurs,*

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

***A** Dida et toute sa famille,*

***A** ma binôme Sarah,*

Enfin, ***A** tous mes amis.*

Abbes

Table des matières

Liste des tableaux

Table des figures

Liste des abréviations

ACO	**A**nt **C**olony **O**ptimisation
ADSL	**A**symmetric **D**igital **S**ubscriber **L**ine
AG	**A**lgorithme **G**énétique
AGoSC	**A**bstract **G**raph **o**f **S**ervice **C**omposition
AS	**A**bstract **S**ervice
ASD	**A**bstract **S**ervice **D**irectory
BPA	**B**usiness **P**rocess **A**gent
BPEL	**B**usiness **P**rocess **E**xecution **L**anguage
CS	**C**oncret **S**ervice
CSD	**C**oncret **S**ervice **D**irectory
CSP	**C**onstraint **S**atisfaction **P**roblem
GPRS	**G**eneral **P**acket **R**adio **S**ervice
GSM	**G**lobal **S**ystem for **M**obile Communications
IA	**I**ntelligence **A**rtificielle
MIP	**M**ixed **I**nteger **P**rogramming
PC	**P**ersonel **C**omputer
PDA	**P**ersonal **D**igital **A**ssistant
QoC	**Q**uality **o**f **C**ontext
QoE	**Q**uality **o**f **E**xperience
QoS	**Q**uality **o**f **S**ervice
RFID	**R**adio **F**requency **ID**entification
SMA	**S**ystÃ¨mes **M**ulti **A**gents
SOA	**S**ervice **O**riented **A**rchitecture
SOAP	**S**imple **O**bject **A**ccess **P**rotocol
SOC	**S**ervice **O**riented **C**omputing
SPSE	**S**ervice **P**roviders **S**earch **E**ngine
SubGoSC	**S**ubgraph of **S**ervice **C**omposition
UDDI	**U**niversal **D**escription **D**iscovery and **I**ntegration
VCL	**V**ienna **C**omposition **L**anguage
VRESCo	**V**ienna **R**untime **E**nvironment for **S**ervice-oriented **Co**mputing
WWF	**W**indows **W**orkflow **F**oundation

Introduction générale

Mark Weiser a présenté sa vision futuriste de l'informatique du XXI^{e} siècle en définissant son principe le "n'importe où" et "n'importe quand" (ubiquitous computing) [1], cela veut dire accéder à l'information de n'importe quel endroit, à tout moment et selon une multitude de modes d'interaction et de média. Les environnements ubiquitaires sont des environnements saturés de dispositifs interconnectés fournissant un accès à une multitude de fonctionnalités qui nous aident dans notre vie quotidienne d'une manière transparente. L'objectif de l'informatique ubiquitaire est de mettre les nouvelles technologies au service de l'utilisateur afin de lui permettre de profiter de l'accés aux différentes fonctionnalités offertes par des différents dispositifs, indépendamment de l'équipement utilisé. La particularité de ces systèmes réside dans leur capacité à adapter continuellement et automatiquement le même service, aux différents contextes et besoins exprimés explicitement ou implicitement par les utilisateurs.

Devant cette panoplie d'équipements mis à notre disposition, de nouveaux besoins surgissent et des défis des environnements ubiquitaires apparaissent aussi. Certaines réponses à ces défis sont aujourd'hui offertes par de nouveaux intergiciels qui permettent de masquer l'hétérogénéité et la distribution des différentes entités des environnements ambiants. Les besoins des utilisateurs nécessitent parfois plus d'un service pour les satisfaire, alors une composition de services est nécessaire. Elle permet de fournir des fonctionnalités qu'un service atomique ne peut offrir seul. La composition de services permet donc à l'utilisateur d'employer les services disponibles dans son environnement pour résoudre des requêtes complexes, tout en considérant les propriétés non fonctionnelles des services et les informations contextuelles pour répondre aux mieux à l'utilisateur. De ce fait, la composition de services est un processus complexe faisant intervenir un ensemble d'étapes intermédiaires telle que la sélection pour pouvoir choisir le meilleur service qui participera à la composition, ou sélectionner le meilleur service composite.

Pour faire face aux problèmes posés par la composition on doit fixer des contraintes de sélection (locale et globale) afin de permettre au système de composition d'élire le service le plus approprié. Plus précisément, la sélection de services compare les services fonctionnellement équivalents en se basant sur leurs performances non-fonctionnelles comme par exemple, le temps d'exécution, la disponibilité, la fiabilité, le niveau d'énergie, le temps de réponse, etc.

Ce travail traite la problématique de sélection et de composition automatique et dynamique de services en environnements ubiquitaires. La composition de service est une tâche délicate car elle ne consiste pas seulement en un assemblage fonctionnel de services mais aussi d'un assemblage non fonctionnel. De ce fait, le respect des contraintes globale et locale de QoS imposées par l'utilisateur est d'une importance capitale dans le processus de composition. Ajoutant à ceci l'adaptation au contexte de services dans tels environnements.

Ce mémoire, articulé autour de quatre chapitres, est structuré de la manière suivante :

Dans le chapitre 1, après avoir défini les environnements ubiquitaires et l'architecture de services ubiquitaires, nous présentons le contexte et une de ses classifications.

Dans le chapitre 2, nous donnons quelques concepts liés à la composition de services, puis nous élaborons une étude critique de certains travaux existant dans la littérature.

Dans le chapitre 3, nous présentons les algorithmes génétiques, puis nous détaillons notre contribution qui consiste en une approche de sélection globale pour la composition de services. La solution proposée est baptisée QSCGA (QoS-awar Service Composition based Genetic Algorithm). Un scénario d'application (assistance de patients dans un hôpital intelligent) illustrant la composition de services est également présenté dans ce chapitre.

Le chapitre 4, quant à lui, porte sur la validation et l'évaluation de performances de l'algorithme proposé et ce en comparant ses résultats avec ceux d'autres algorithmes proposés dans la littérature.

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

1

Présentation des systèmes ubiquitaires

1.1 Introduction

Les téchnologies de l'informatique et des réseaux sont en perpétuelle évolution ce qui fait que les capacités de traitement et de communication de l'information croîtent au même rythme que la miniaturisation des supports. La mobilité et la reconfiguration dynamique seront des traits dominants de ces systèmes, imposant ainsi une adaptation permanente des applications.

Ce chapitre fera l'objet d'une présentation générale de l'informatique ubiquitaire qui constitue le cadre général de notre travail, ses différents aspects et sa relation avec les autres domaines de l'informatique. Nous utiliserons des scénarios pour mettre en évidence l'un des aspects les plus novateurs de l'informatique ubiquitaire qui est l'assistance implicite et discrète d'un utilisateur dans les tâches qu'il accomplit au quotidien.

1.2 Informatique ubiquitaire

1.2.1 Présentation de l'informatique ubiquitaire

En 1991, Mark weiser [1] a présenté sa vision futuriste de l'informatique de XXI^{e} siècle en analysant l'évolution du marché mondial des ordinateurs, et il a constaté que le nombre de microprocesseur par individu est toujours en constante croissance (voir la figure 1).

Cette informatique dite ubiquitaire se focalise sur les interactions des ressources de calculs distribuées, pour assurer une assistance transparente d'un utilisateur dans les tâches qu'il accomplit au quotidien [2, 3].

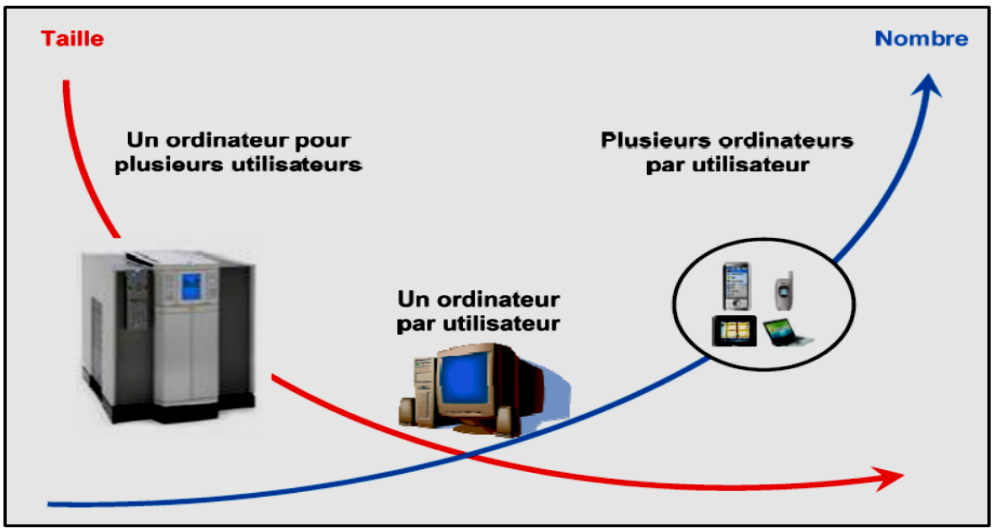


FIGURE 1.1 – Évolution du marché mondial des ordinateurs.

Les systèmes de l'informatique pervasif visent à adapter le comportement des objets communicant en vue de répondre aux besoins des utilisateurs suivant les changements du contexte de l'environnement et de ses composants. Les dispositifs pervasifs établissent une connexion entre eux et peuvent être connectés aux différents types d'appareils capturant les changements dans l'environnement [4]. Ces objets communicants sont des actionneurs agissant sur l'environnement ou des capteurs fournissant de nombreuses informations.

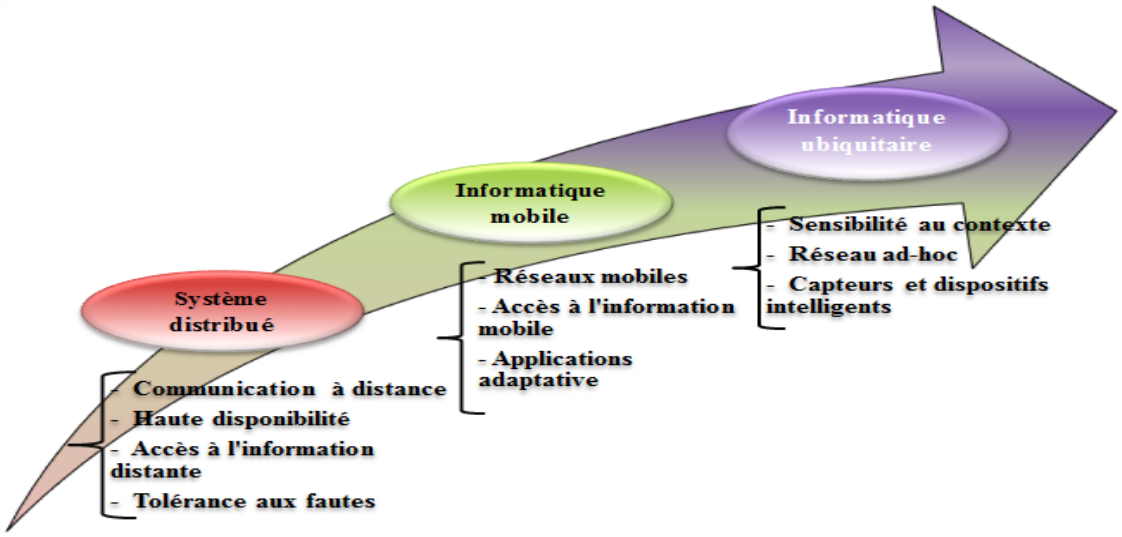


FIGURE 1.2 – Évolution vers l'informatique ubiquitaire.

Pour éviter tout risque de confusion entre les termes utilisés dans le domaine de l'informatique ubiquitaire, nous présentons les définitions suivantes qui le caractérisent [5] :

- **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.

1.2.2 Définition d'un environnement ubiquitaire

L'évolution technologique actuelle rend aujourd'hui la vision de Mark Weiser réaliste, à la suite de l'apparition des nouveaux systèmes embarqués ayant des tailles de plus en plus petites et enfouis dans différents objets de la vie quotidienne.

L'avènement des réseaux de communications sans fil avec les standards de télécommunication, tels que GSM, GPRS, particulièrement utilisés pour la téléphonie mobile, mais également avec l'apparition de Wifi, RFID et Bluetooth pour les équipements informatiques tels que les assistants personnels (PDA : personal digital assistant) et les téléphones cellulaires, a permis à ces équipements de communiquer pour coopérer. Ceci se fait d'une manière transparente pour l'utilisateur sans son intervention en lui offrant la possibilité de se concentrer sur sa tâche principale au lieu de configurer et de gérer l'ensemble

des équipements informatiques mis à sa disposition. L'informatique ubiquitaire favorise par exemple la création d'environnements intelligents tels que la maison intelligente capable de gérer automatiquement les différents équipements présents au domicile de l'utilisateur [8].

En effet l'environnement ubiquitaire est un environnement saturé de dispositifs interconnectés entre eux via différents mode d'interaction (Wifi, Bluetooth, RFID, etc.).

1.2.3 Domaine d'application de l'informatique ubiquitaire

1.2.3.1 Domaine médical

L'informatique ubiquitaire a aussi un effet très important dans le domaine médical que dans d'autres domaines. Elle consiste a accélérer l'intervention de l'équipe médicale, grâce aux notifications qui seront transmises aux personnes concernées. En citant par éxemple le cas de l'hôpital intélligent [4], cité comme exemple de scénario (voir section 1.2.3.1).

1.2.3.2 Domaine de voyage

Dans le domaine de l'information voyageur, un service peut être qualifié d'ambiant, s'il renseigne le voyageur en fonction de son environnement direct. Il s'agit donc de service capable d'adapter l'information en fonction des spécificités de l'utilisateur, de sa localisation selon le besoin afin de l'aider à réaliser son voyage ; il s'agit ici de la sensibilité au contexte (voir section 1.5.3).
Dans cette catégorie, nous pouvons citer par exemple des services aidant les voyageurs handicapés, ou les voyageurs dans un lieu de transit. L'objectif de ces différents systèmes est de confronter des informations sur le voyageur avec une information locale qui peut l'aider dans son déplacement. Ainsi, la personne aveugle prenant le bus est alertée dès que son bus arrive à proximité de son arrêt.

1.2.3.3 Domotique

L'informatique ubiquitaire dans la domotique est utile dans différents cas (assitance des personnes dépendante, détection d'intrusion, etc.) et différents lieux (maisons, hôtels, entreprises, etc.). Généralement l'installation d'une maison intélligente a beaucoups d'avantages dans la vie quotidienne car elle permet de centraliser le contrôle des différents système de la maison (chauffage, porte, fenêtre, etc.) et vise à apporter des solutions pour répondre aux besoins de confort (gestion d'énergie, optimisation de l'éclairage et de la climatisation, etc.), de sécurité (alarme) et de communication (signaux visuels, sonores, etc.).

1.2.3.4 Réseaux sociaux

Lors d'une inscription dans un réseau social ou professionnel (Linked, Google +, etc.), un remplissage du profil est nécessaire ; et en fonction des informations introduites, le système propose tous les utilisateurs ayant un profil similaire, ou dans le cas des réseaux professionnelles toutes les sociétés intéressées par ce genre de profil. L'utilisateur peut être assimilé à une catégorie (débutant, professionnel ou expert), dans ce contexte le système lui suggère des formations professionnelle pour accroitre ses compétences.

1.2.4 Scénarios illustratifs de l'informatique ubiquitaire

Pour rendre clairs les concepts fondamentaux de l'informatique ubiquitaire, nous présentons dans cette section deux scénarios différents qui donnent un aperçu sur les étapes parcourues afin de répondre à la requête de l'utilisateur. Le scénario de recherche d'une conférence [6] met l'accent sur les caractéristiques ubiquitaires des logiciels tandis que le premier scénario [4] se concentre sur les questions relatives au déploiement des dispositifs dans les réseaux dynamiques.

1.2.4.1 Scénario d'un hôpital intelligent

On considère un hôpital intelligent où des patients, des infirmières, des médecins, etc. sont impliqués. Cet hôpital est équipé de technologies (matériel et logiciel) de capteurs de contexte dans les chambres, les couloirs et le jardin à la disposition des personnes concernées. Un système de l'informatique pervasif réactif au contexte adapté pour la surveillance et le suivi des patients dans les hôpitaux aide à minimiser l'engagement des spécialistes aux activités moins importantes. L'intervention humaine pourrait être nécessaire uniquement lors d'une alerte par le système.

La Figure 3 illustre un cas d'utilisation spécifique du scénario de l'hôpital intelligent [4]. Un matin, le docteur Pascal et ses collègues sont en réunion de consultation hebdomadaire (1). Michel, le patient, est dans un jardin pour profiter du soleil matinal (2). Soudain, Michel se sent épuiser et tombe par terre (3). Le portable et le badge qu'il porte sur son corps livrent immédiatement toutes les informations nécessaires au dispositif informatique se trouvant à proximité (4). Ensuite, le système envoie (5) un message d'alerte (4) à Ada, l'infirmière sur son téléphone intelligent. Un autre message envoyé à la salle de serveur central dans le bureau de la secrétaire médicale à des fins administratives et rediffusion (7). À partir de l'agenda de Dr Pascal, le système extrait l'information que le médecin est

en ce moment en réunion, le système l'informe (8) par message SMS sur son téléphone cellulaire qui clignote en rouge à la réception de ce type de message d'urgence dans une réunion. La camera face au jardin (9), où Michel est situé, est activé et son image est envoyée (10) à l'ordinateur central pour l'adapter et la diffuser (11, 12, 13) à tous les terminaux concernés (14, 15, 16). Les secouristes (17) sont également informés (18) de la situation. A la suite, Michel est amené (19, 20) au chambre de traitement (21) par les secouristes d'urgence. Le docteur Pascal, qui a déjà été au courant de la situation actuelle de son patient, a fait toutes les consultations nécessaires (22, 23, 24), le patient est déjà dans la salle de traitement et le médecin a ordonné les médicaments appropriés à préparer(25). Ada est également dans la salle de traitement (26) pour fournir l'aide nécessaire au patient.

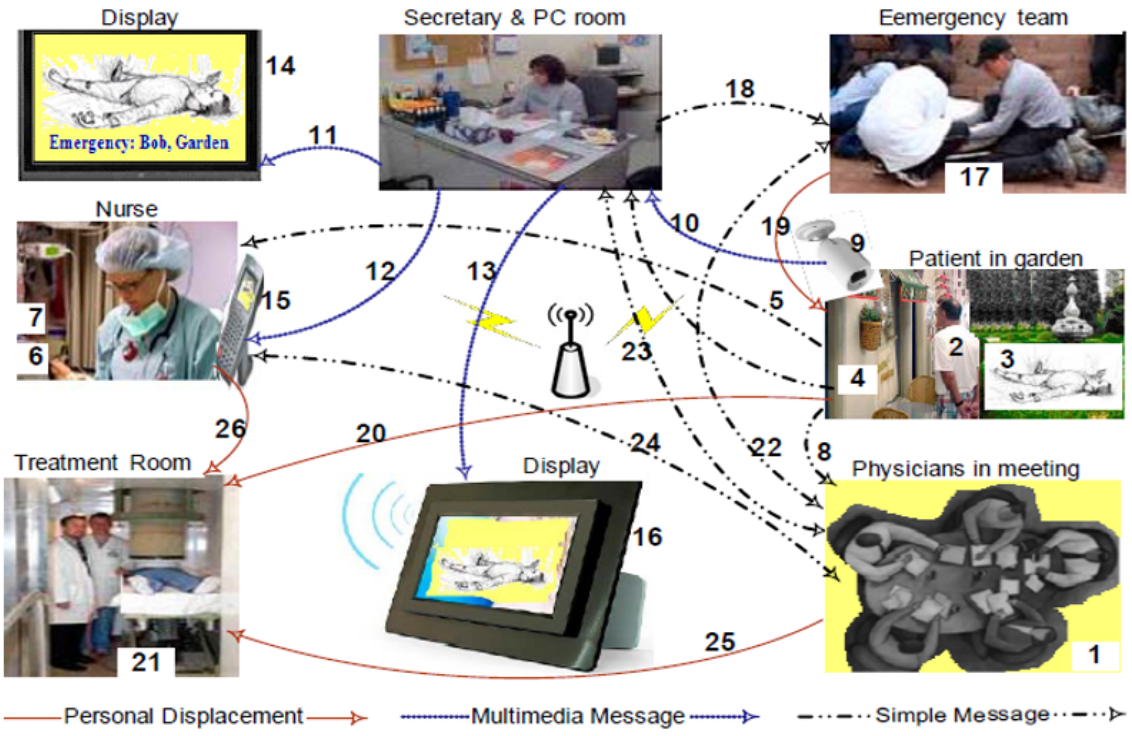


FIGURE 1.3 – Le scénario illustrant l'hôpital intelligent.

1.2.4.2 Scénario de la recherche d'une conférence

Les chercheurs cherchent d'éventuels appels à communication susceptibles d'être pertinents avec leurs domaines de recherche [6]. L'utilisateur envoie une requête de recherche de conférence (CherchConf) avec les paramètres contenant toutes les exigences sur la conférence (DonnéesConf) tels que le thème, la date limite de soumission, l'éditeur, etc. Le service Recherche reçoit la requête et génère une nou-

velle requête ConfRequete(DonnéesConf) auprès du service Choréographeur. Ce dernier lance une première recherche auprès du registre en invoquant l'opération ConfRequete-Reg(DonnéesBasiqueConf) où DonnéesBasiqueConf représente les données de base d'une conférence. A partir du résultat de la dernière opération, le service Choréographeur invoque la méthode ConfRequeteFourn(DonnéesConf) du service FournisseurInfoConf qui retourne la liste de toutes les conférences répondant aux critères de recherche déjà définis. L'extraction de cette liste se fait suite à l'accès à la base de données qui contient la liste des conférences.

1.2.5 Défis et caractéristiques de l'informatique pervasive

Les critères de nombre, mais surtout de variété et de dynamicité des équipements informatiques impliqués dans une application décident du caractère pervasif de celle-ci. L'internet, les réseaux locaux comme le réseau domestique et les réseaux mobiles peuvent être considérés comme des environnements pervasifs dès lors que les applications envisagées impliquent l'interaction d'équipements informatiques variés et à la présence variable [9].

L'ouverture de l'environnement ubiquitaire à des entités électroniques distinctes et variées dans un contexte évoluant rapidement pose les défis suivant :

1.2.5.1 Invisibilité et transparence

Un environnement ambiant doit fournir une connectivité globale et s'affranchir de sa nature informatique. Un des premiers objectifs des environnements ambiants est donc de généraliser et banaliser les réseaux de communications sans fil et de rendre toute la technologie sous-jacente invisible à l'utilisateur. Un deuxième objectif est de lui rendre l'accès à l'information plus transparent vis à vis du lieu et du contexte actuel. Pour cela le système doit nécessiter un minimum d'intervention humaine [7].

1.2.5.2 Communication distante avec des entités inconnues

L'organisation des entités distinctes en un réseau cohérent pose le premier défi de la répartition des équipements. Un système réparti coordonne les fonctions de plusieurs unités de calculs disposées en réseau afin de fournir les fonctionnalités agrégées d'une même application. Des techniques intergicielles permettent de masquer l'aspect distribué de fonctions distantes dans les langages de programmation [10].

1.2.5.3 Hétérogénéité

Un environnement ubiquitaire est chargé des équipements, qui sont très variés (ordinateur portable, PDA, Smartphone, appareil médical, etc.). Ces dispositifs fournissent de différents services fonctionnant à base de différents systèmes qui diffèrent généralement dans leurs configurations matérielles et logicielles. Ces équipements se communiquent via divers technologies de communication filaires ou non filaires (ADSL, Wifi, Bluetooth, etc.).

1.2.5.4 Dynamicité

La nature dynamique des environnements ubiquitairs est due aux changements fréquents qui se produisent dans les environnements ambiants alors que la nature incertaine est due à l'apparition aléatoire de ces changements [7]. La dynamicité des environnements pervasifs est l'une de ses caractéristiques, pour cela l'exigence d'adaptation des applications à un contexte évoluant rapidement est nécessaire. Pour cela on constate deux type de dynamicité [11] :

- **Objets communicants** : Les ressources limitées pour les objtes communicants, rendent leurs disponibilités variées au cours du temps à cause de leurs natures mobiles, pour cela, ces objets ont des ressources limitées, des problèmes matériels ou des bogues logiciels les rendant inaccessibles, ou bien sont ajoutées ou supprimées des environnements physiques pour des raisons de maintenance.

- **Contexte** : Les systèmes ubiquitaires permettent l'automatisation de tâches attribuées aux utilisateurs. À l'instar d'un utilisateur, les systèmes ubiquitaires doivent être sensibles à leur environnement physique pour prendre les décisions appropriées : adapter les services offert à l'utilisateur en fonction de son activité et de sa localisation, adapter la demande de ressources selon les capacités et la disponibilité des équipements, enfin adapter la réponse du système en fonction des variables de l'espace environnant les équipements et les utilisateurs.

1.2.5.5 Fiabilité

La fiablité des systèmes d'informatique ubiquitaire est essentielle, car un mauvais fonctionnement pourrait entraîner des conséquences néfastes sur les activité de l'utilisateur, qui est au centre de ces systèmes. Par exemple, dans le cas d'assistance à une personne malade, si un capteur (ex. de chute) tombe en panne, la sécurité de la personne pourrait

être compromise. De plus, chaque domaine d'application possède des propriétés critiques spécifiques, que le système déployé doit garantir [12].

1.3 Les intergiciels pour l'environnement ubiquitaire

Le terme "intergiciel" ou "middleware" est apparu pour la première fois dans le cadre d'outils logiciels pour des applications réparties. L'intergiciel représente la couche logicielle entre les couches matérielle et l'application logicielle finale. Il gère les communications entre les entités réparties [13].

Un intergiciel se situe au-dessus du système d'exploitation (OS pour Operating System) et en dessous des applications (c'est-à-dire des clients et/ou des services) de son hôte [8].

L'objectif principal des intergiciels est de masquer les aspects complexes de l'imbrication logicielle nécessaire aux applications complexes, et de fournir des fonctions complémentaires à la fonction principale d'un logiciel. Ces fonctions sont généralement appelées préoccupations non fonctionnelles, les aspects fonctionnels désignant le logique métier de l'application [10].

Quel que soit le type des objets communicants (mobile ou statique, filaire ou non) sur lesquels un intergiciel est déployé il doit fournir les fonctionnalités/services élémentaires suivants [9] :

- **Protocole d'interaction :** les clients et les services étant distribués sur des hôtes différents. L'objet client doit avoir une référence d'objet pour l'objet serveur.

- **Fiabilité :** l'un des rôles de l'intergiciel est de prendre en charge une détection des erreurs et/ou des mécanismes de corrections supplémentaires ou complémentaires selon les cas pour pallier au manque de fiabilité de certains protocoles.

- **Hétérogénéité :** l'intergiciel prend en charge l'hétérogénéité de différentes façons. CORBA (Common Object Request Broker Architecture) et COM (Component Object Model) ont tous les deux multiples liaisons de langages de programmation pour que les objets client et serveur n'ont pas besoin d'être écrit dans le même langage de programmation.

– **Coordination :** Les primitives de synchronisation par défaut dans l'objet middleware sont des requêtes synchrones, qui bloquent l'objet client jusqu'à ce que l'objet serveur renvoie la réponse.

1.4 L'architecture Orientée Service (SOA)

Une architecture de services permet de standardiser l'accès aux ressources et/ou aux fonctionnalités des objets communicants en les représentants sous formes de services [8]. SOA est une architecture de conception qui est définie par un ensemble de principes de conception [14]. Un service est défini par un contrat (appelé aussi interface) qui est une spécification abstraite de ses fonctionnalités. Ce contrat décrit : (i) ce que le service fournit, (ii) comment y accéder et (iii) éventuellement, quelles sont ses propriétés non fonctionnelles.

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

Les interactions entre clients et services se font en plusieurs étapes (voir la Figure 4) :

- Le fournisseur de services publie la description de ses services auprès du service de découverte (étape1).

- Le consommateur de services interroge le service de découverte en lui soumettant la description (partielle) du ou des services requis (étape 2).

- Le service de découverte renvoie le contrat du service et la référence d'une ou plusieurs instances de services correspondant (étape 2).

- Le consommateur de services initie les interactions avec le fournisseur de service suivant les termes du contrat du service (étape 3).

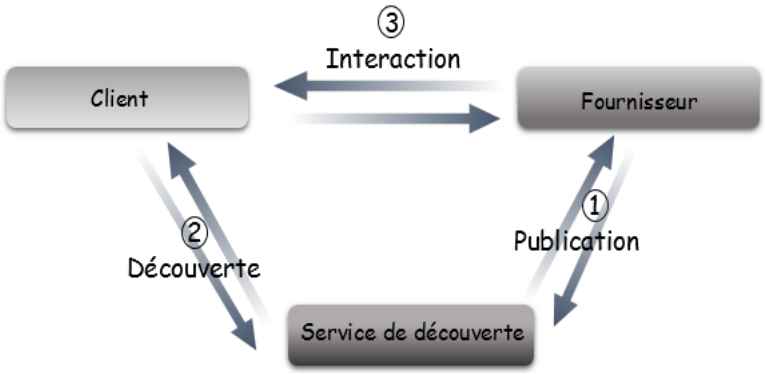


FIGURE 1.4 – Architecture de services ubiquitaires.

1.5 La notion de contexte

Le concept de contexte joue un rôle clé dans les systèmes ubiquitaires. Plusieurs auteurs ont essayé de cerner ce concept au moyen de définition et de classifications, dont nous citons les plus importantes dans cette section. Nous nous intéressons également à la sensibilité au contexte, pour présenter quelques solutions qui prennent ces concept en considération dans la composition de services.

1.5.1 Définition

Parmi les premiers à essayer de définir le contexte, se trouvent Schilit et Theimer, pour lesquels le contexte est constitué de la localisation de l'utilisateur, ainsi que des identités et des états des personnes et des objets qui l'entourent [15]. Brown et al. [16] ajoutent à cette définition des données telles que l'identité de l'utilisateur, son orientation ou la température. Ryan et al [17] ajoutent la notion de temps.

Pascoe [15] introduit un élément important : l'intérêt. En effet, il définit le contexte comme un sous-ensemble d'états physiques et conceptuels qui ont un certain intérêt pour une entité donnée.

Cette notion d'intérêt ou de pertinence est reprise par Abowd et al. [18] dans leur définition, qui est communément acceptée : "Le contexte couvre toutes les informations qui peuvent être utilisées pour caractériser la situation d'une entité. Une entité est une personne, un endroit ou un objet que l'on considère pertinent par rapport à l'interaction entre un utilisateur et une application, y compris l'utilisateur et l'application eux-mêmes.".

Dans [19] ont conclu que la plupart des définitions sont des réponses aux questions suivantes :

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

1.5.2 Catégorisation du contexte

Plusieurs catégorisations ont été proposées (ex. en deux classes : le contexte primaire qui contient les informations sur la localisation, l'identité, le temps et l'activité (statut) ; le contexte secondaire qui peut être déduit de ce dernier), on cite la catégorisation faite dans [20]. Elle comprend les cinqs classes suivantes :

1. **Contexte utilisateur :** permet d'obtenir les informations sur les utilisateurs du système informatique (son identification, ses relations avec les autres usagers, la liste de ses tâches, etc.).

2. **Contexte physique :** offre la possibilité d'intégrer des informations relatives à l'environnement physique, telles que la localisation, l'humidité, la température, etc.

3. **Contexte du réseau :** fournit aussi des informations de l'environnement, mais ces dernières se rapportent essentiellement au réseau informatique (connectivité, bande passante, protocole, etc.).

4. **Contexte d'activité :** répertorie les évènements qui se sont déroulés dans l'environnement ainsi que leur estampille temporelle (entrée d'une personne et le moment de son entrée, tempête de neige et l'estampille temporelle de la tempête, etc.)

5. **Contexte matériel :** permet d'identifier les appareils de l'environnement qui peuvent être utilisés. Il inclut le profil et les activités des dispositifs de l'environnement (identification, localisation, niveau de la batterie, etc.).

1.5.3 Sensibilité au contexte

Le terme "sensible au contexte" a été introduit la première fois dans le cadre de l'informatique mobile par Schilit et Theimer [15]. L'idée de sensibilité au contexte émerge naturellement du concept d'informatique ubiquitaire.

En effet, le fait de disposer d'informations contextuelles sert à adapter l'application au contexte perçu afin de mieux répondre aux attentes des utilisateurs. L'idée principale de l'informatique sensible au contexte est de sortir le plus possible les humains de la boucle de contrôle des machines, en réduisant les interactions entre l'humain et la machine.

La notion de sensibilité au contexte (context-awareness) a donc été produite dans le cadre des recherches sur l'informatique ubiquitaire. La sensibilité au contexte s'agit de la capacité d'un système à découvrir et à réagir à des changements dans l'environnement où il se trouve. Ils signalent également l'importance de l'adaptation du système à ces changements [15].

Pour Abowd et al. [18], un système est sensible au contexte s'il utilise celui-ci pour fournir à l'utilisateur des informations et des services pertinents. Cette pertinence dépend de l'activité que l'utilisateur est en train de réaliser. Ces auteurs proposent également une classification des systèmes sensibles au contexte selon leur capacité de réponse aux changements de contexte. Cette réponse peut soit présenter des informations ou des services à l'utilisateur, exécuter automatiquement un service, ou stocker des données contextuelles.

1.6 Conclusion

L'informatique ubiquitaire constitue la pierre angulaire de l'informatique future. Ce domaine combinant les aspects de l'informatique distribuée et de l'informatique mobile, adopte une nouvelle vision des équipements et des applications [21]. L'un des aspects contribuant à la réalisation de cette notion d'informatique ubiquitaire est de permettre à ces systèmes de s'adapter pro-activement aux changements du contexte des applications

et de celui de l'utilisateur.

Le chapitre suivant fera l'objet de présenter quelques solutions proposées pour la composition de service dans les environnements ubiquitaires, afin de remédier quelques limitations.

2

Composition de services. Définitions et état de l'art

2.1 Introduction

Quand la requête de l'utilisateur est non satisfaite par un seul service, il est inévitable de composer les services atomiques de l'environnement pour pouvoir satisfaire la demande du client. Pour cette raison, plusieurs services peuvent interagir et échanger dynamiquement des informations. L'objectif de cette interaction est l'accomplissement d'un but complexe non concevable par un seul service. Cette agrégation des compétences pour réaliser un but commun est connue par "composition de services". En parlant d'une composition de services, le problème qui se pose est "comment sélectionner et composer ces services ?".

Dans cette partie nous présentons des définitions préliminaires, ensuite nous détaillons les différents types de méthodes de composition de services présents dans la littérature. Enfin un état de l'art des travaux qui proposent des approches de composition de service est décrit suivi d'une étude comparative des différentes solutions.

2.2 Présentation de la composition de services

2.2.1 Définitions préliminaires

2.2.1.1 Définition de service

La plus simple des définitions qu'on puisse trouver est celle du dictionnaire : "une action réalisée par une entité pour le compte d'une autre", ou "Un service est une unité de comportement définie selon un contrat entre le client et le fournisseur de service".
Selon Papazoglou : "Services are self-describing, open components that support rapid, low-cost composition of distributed applications" [22]. Cette définition considère le service comme une entité qui peut être utilisée à travers sa description.
Avant d'utiliser un service, il est nécessaire de connaitre ses capacités fonctionnelles (quelle est exactement l'action que le service peut réaliser) et ses caractéristiques non-fonctionnelles (la disponibilité, le coût de son utilisation, etc.) [23].

2.2.1.2 Définition de la composition de services

La composition de services peut être vue comme un mécanisme permettant l'intégration des services pour réaliser une application. Le résultat d'une composition peut être un nouveau service, appelé service composite [24].

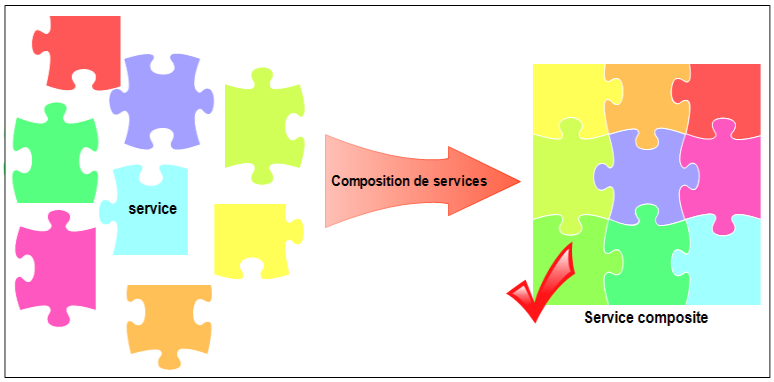


FIGURE 2.1 – Composition de services.

La composition de services fournit les mécanismes nécessaires pour assembler des services dans un environnement orienté services. En d'autre terme, la composition de service est l'opération appliquée sur un ensemble de services qui ne peuvent satisfaire la requête de l'utilisateur d'une façon atomique, mais d'une façon complémentaire, pour avoir un service composite afin de répondre aux exigences du client.

Cependant, pour passer d'un ensemble de services à une composition de services correctement structurée, il faut suivre un certain nombre d'étapes de la spécification à la composition concrète exécutable [24] :

- ***La définition de l'architecture fonctionnelle :*** cette phase est faite pour identifier les fonctionnalités attendues de l'application résultant de la composition de services ;
- ***L'identification des services :*** selon les fonctionnalités attendues, on détermine les services nécessaires à la composition ;
- ***La sélection des services et leur implantation :*** à partir des services identifiés à l'étape précédente, il faut sélectionner les services qui répondent correctement aux besoins adaptés ;
- ***La médiation entre services :*** la médiation de services a pour objectif de résoudre les hétérogénéités présentes entre services afin de permettre des interactions réussies, la médiation consiste à résoudre les conflits entre deux acteurs. Cette tache est effectuéee par un élément spécifique appelé médiateur ;
- ***Le déploiement et l'invocation des services :*** une fois la composition est correctement réalisée, il faut déployer les services sur les plates-formes d'exécution.

2.2.2 Séléction de services avec QoS

La sélection de services est la décision de choisir des services en particulier parmi un ensemble de services sur la base de besoins fonctionnels ou des paramètres de QoS [23].

Deux types de sélection de services existent, une sélection locale et une sélection globale [25].

2.2.2.1 Sélection locale

La sélection locale de services consiste à trouver pour chaque tâche, indépendamment des autres, le service le plus approprié. Si la recherche locale permet de prendre en compte des exigences fines de QoS sur les services de la composition, elle ne permet pas d'optimiser la QoS globale et ne peut pas considérer de contraintes sur la globalité de la composition.

2.2.2.2 Sélection globale

La sélection globale de services tend à résoudre les limitations de la sélection locale. A l'instar de la sélection locale, une prise de décision multicritères est appliquée sur le

résultat composite pour décider des services permettant d'atteindre une QoS optimale.

La recherche globale permet d'optimiser (maximiser ou minimiser) des critères de QoS sur le résultat (service composite) mais ne permet pas de spécifier des contraintes à une échelle intermédiaire ou locale.

2.3 Méthodes de composition de services

Dans les différentes solutions proposées pour le problème de composition de services, les auteurs se focalisent sur différentes méthodes (dynamique, statique, manuelle, automatique). Dans ce qui suit nous définissons chacune de ces méthodes.

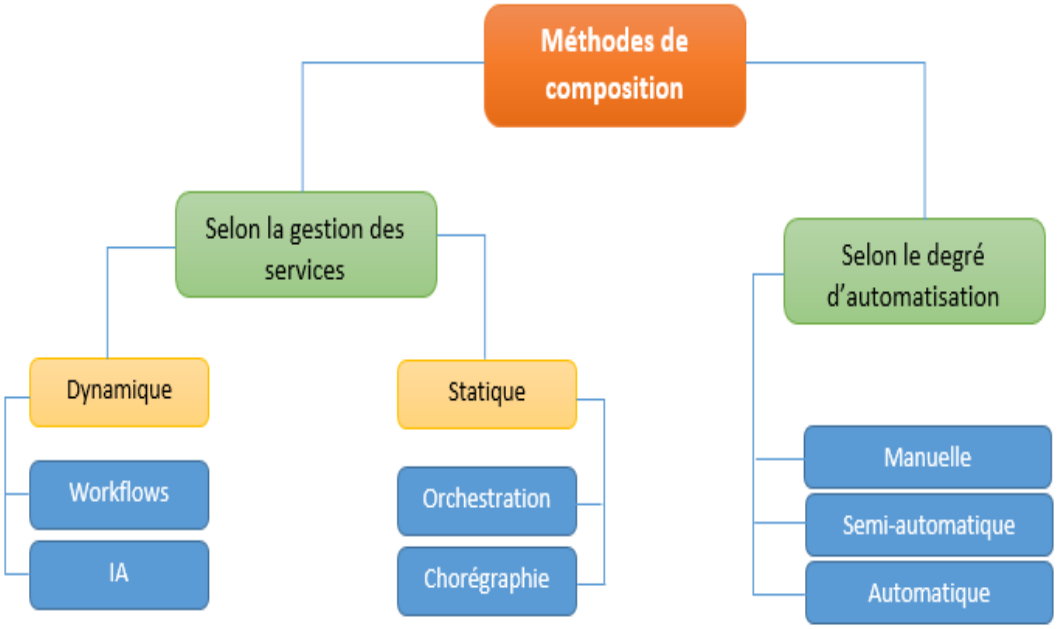


FIGURE 2.2 – Classification des méthodes de composition de services.

Les solutions proposées pour la composition de services peuvent être classifiées selon deux axes [26] : soit en fonction du degré de participation de l'utilisateur dans la définition du schéma de composition, et dans ce cas ces approches peuvent être manuelles, semi-automatiques ou automatiques. Ou bien selon que la sélection des services et la gestion du flot soient faites a priori ou non, dans ce cas l'approche sera dite statique ou dynamique.

2.3.1 Classification selon le degré d'automatisation

Selon le degré d'automatisation de la composition, cette dernière peut être faite de trois manières différentes [26] :

2.3.1.1 La composition manuelle

La composition manuelle suppose que l'utilisateur génère la composition textuellement et sans l'aide d'outils dédiés. Ainsi, l'utilisateur se charge de définir son besoin en termes de composition des services en consultant un registre de services et en se basant essentiellement sur sa connaissance sur le domaine.

2.3.1.2 La composition semi-automatique

Les techniques de composition semi-automatiques sont un pas en avant en comparaison avec la composition manuelle, dans la mesure où elles font des suggestions sémantiques pour aider à la sélection des services dans le processus de composition

2.3.1.3 La composition automatique

La composition automatique prend en charge tout le processus de composition et le réalise automatiquement, sans qu'aucune intervention de l'utilisateur ne soit requise.

2.3.2 Classification selon la géstion des services

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

2.3.2.1 Composition statique (par procédés)

On distingue deux types de composition par procédés : orchestration et chorégraphie.

a) Orchestration : l'orchestration des services permet de décrire l'enchaînement des services selon un canevas prédéfini, et de les exécuter. Généralement on parle d'orchestration quand il y a un processus principal (l'orchestrateur) qui prend le contrôle du déroulement de la composition et coordonne les différentes opérations des différents services [26].

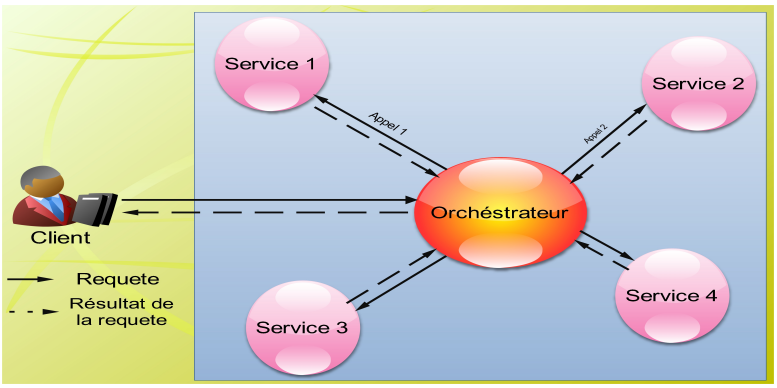


FIGURE 2.3 – Vue générale de l'orchéstration.

L'orchestration correspond à un schéma de collaboration centralisée (voir la figure. 2.3) où une seule entité réunit l'ensemble des appels aux différents services de la composition. Cette entité invoque les services dans un ordre prédéfini [27].

b) Chorégraphie : une chorégraphie décrit le flux de messages échangés par un service lors de son interaction avec d'autres services (voir la figure. 2.4). Il s'agit donc d'une manière décentralisée de gérer une composition puisque chaque service, acteur, est responsable d'une partie du workfow [25].

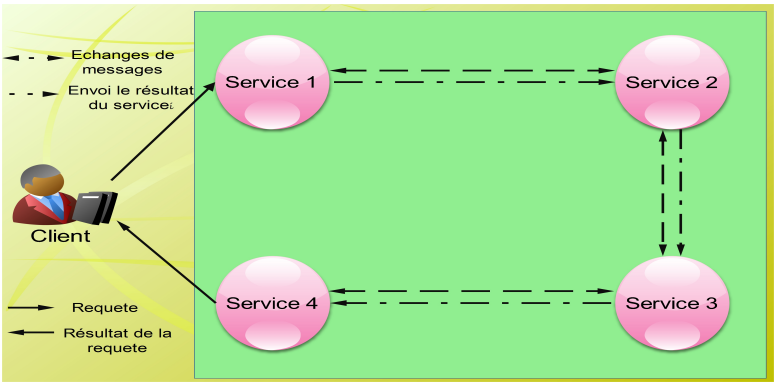


FIGURE 2.4 – Vue générale de la chorégraphie.

Dans une chorégraphie, à chaque pas de l'exécution (i.e. à chaque étape de la composition), un service choisit le service qui lui succède et implémente ainsi une partie de la chorégraphie. La composition de type chorégraphie n'est pas connue, ni décrite à l'avance [28].

2.3.2.2 Composition dynamique

La composition dynamique prend en compte les services disponibles, leurs fonctionnalités et le but à atteindre que ce soit avant ou pendant l'exécution des services.

Benatallah et al. considèrent dans [29] la composition dynamique comme l'agrégation de services permettant de résoudre un objectif précis soumis par un utilisateur en prenant en compte ses préférences. Etant donné une spécification de haut niveau des objectifs d'une tâche particulière, la composition dynamique implique la capacité de sélectionner, de composer et de faire interopérer des services existants.

Les différentes approches existantes pour la composition dynamique de services peuvent être regroupées en deux courants, les approches basées sur les workfows et les approches basées sur les techniques de l'intelligence matricielle [26] :

a) Les approches orientées Workflow

La composition en workflows vise à créer des services de plus grande valeur ajoutée réalisant des fonctions plus complexes. Les workflows sont particulièrement utilisés pour décrire comment des services ou activités "informatiques" sont connectés pour construire des procédés métiers complexes [30].

Un workflow est composé d'un nombre de tâches ou d'activités, de dépendances entre tâches, de règles et de participants. Dans un workflow, une tâche peut représenter une activité humaine ou un système (logiciel) [31].

Du coté des compositions dynamiques, le modèle du processus ainsi que la sélection du service sont faits automatiquement. A cet égard, une composition automatique demande des workflows capables de reconnaitre les services correspondants à chaque tâche, mais aussi de trouver d'autres services au cas où ceux-ci soient indisponibles ou dévient de leur exécution normale [31].

b) Approches issues de l'intelligence artificielle (IA)

La composition de services par des techniques issues de l'intelligence artificielle, et plus particulièrement par des techniques à base de règles, est la voie qui semble prometteuse

pour certains auteurs comme [32].

Dans ce qui suit, nous présentons quelques axes de recherche concernant la composition par planification, par SMA et par d'autres techniques d'intelligence artificielle.

✓ ***Calcul situationnel***

Le problème de la composition est abordé de la façon suivante : la requête de l'utilisateur et les contraintes des services sont représentées en termes de prédicats du premier ordre dans le langage de calcul situationnel. Les services sont transformés en actions (primitives ou complexes) dans le même langage. Puis, à l'aide de règles de déduction et de contraintes, des modèles sont ainsi générés et sont instanciés à l'exécution à partir des préférences de l'utilisateur [33].

Le monde est conçu comme un arbre de situations, débutant par la situation initiale S_0, et évoluant jusqu'à la nouvelle situation par l'application d'une ou plusieurs actions. Une situation s donnée correspond toujours à un historique de l'ensemble d'actions réalisées sur S_0 [31].

✓ ***La planification***

La planification est l'un des domaines de l'Intelligence Artificielle (IA) qui permet de choisir et d'organiser des actions, en fonction d'un but donné. Le mot planification est également utilisé dans plusieurs autres domaines, tels que l'économie, la politique, l'architecture et encore dans la vie de tous les jours [31].

La planification en IA peut aussi être conceptualisée comme un choix et une organisation d'actions pour changer l'état d'un système. Par extension, elle produit un plan qui est présenté sous la forme d'une collection de descriptions d'opérateurs. Le planificateur ou générateur de plans est le système qui produit un plan. L'exécution est la réalisation des actions présentes dans le plan. L'exécution d'un plan modifie les propriétés du monde en le faisant évoluer de l'état initial jusqu'au but désiré [31].

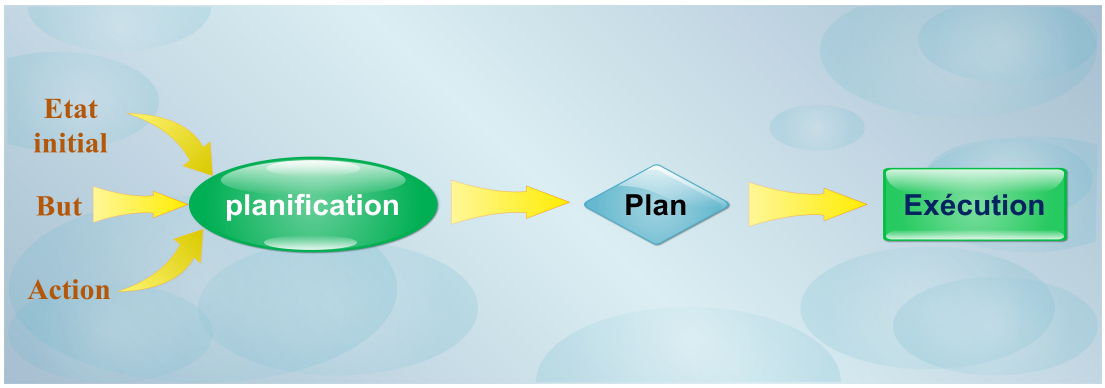


FIGURE 2.5 – Méthode de composition de services par planification.

✓ ***Preuves par théorèmes***

Dans cette approche, les services disponibles et les requêtes utilisateur sont traduites dans un langage du premier ordre. Puis des preuves sont produites à partir d'un prouveur de théorèmes [33]. La requête de l'utilisateur est décrite comme un théorème que l'on souhaite prouver [31].

✓***Composition avec SMA***

La composition de services peut être implémentée aussi en utilisant des SMA (Systèmes Multi Agents). Dans cette approche, chaque agent présente un service et sert à satisfaire une partie de la requête de l'utilisateur en utilisant ses propres capacités [33].

2.4 Défis de la composition de services

2.4.1 Respect des contraintes de QoS globale de l'utilisateur

Une exigence clé dans la composition de services est de répondre à des tâches complexes requises, tout en respectant les contraintes imposées par les utilisateurs sur la qualité de service (QoS).

Un grand nombre de services peut être trouvé pour la réalisation de chaque sous-tâche d'une tâche complexe, le problème est la sélection du meilleur ensemble de services (à savoir en termes de qualité de service) pour participer à la composition, répondant aux exigences de qualité de service globale de l'utilisateur.

2.4.2 Limite de temps de sélection de services

La sélection de service et la composition doivent être effectuées lors de l'exécution. Par conséquent, le temps d'exécution des algorithmes de sélection de services est fortement limité alors que la complexité de calcul du problème est NP-difficile.

2.4.3 La dynamicité

Les environnements dynamiques exigent de répondre à la volée aux requêtes utilisateurs (c.à.d, au moment de l'exécution) et que la disponibilité de services ne peut pas être connu a priori.

La fluctuation des de qualité de service en raison de la dynamicité de l'environnement. Ce probleme se pose par exemple quand un ou plusieurs services fesant partie d'une composition ne sont plus disponibles ou leur QoS baisse (par exemple, en raison de la deconnexion du reseau ou la connectivite reseau faible) pendant l'execution de la composition.

2.4.4 La dégradation de la QoS

Un service qui satesfait les contrainte de QoS est sélectionné pour participer à une composition peut ne pas offrir les mêmes QoS lors de l'invoquation.

2.5 Etude de quelques traveaux de composition de services en environnements ubiquitaires

2.5.1 QoS-aware Service Composition in Dynamic Service Oriented Environments

2.5.1.1 Aperçu de l'approche

N. Ben Mabrouk et al. [34] proposent un algorithme efficace de sélection de services, qui fournit le terrain adéquat pour la composition de services avec QoS dans des environnements dynamiques de services.

• ***La découverte :*** Les auteurs supposent que l'utilisateur accède à une interface "utilisateur graphique" pour formuler sa demande. Cette interface guide l'utilisateur à exprimer sa requête, en termes d'exigences fonctionnelles et non-fonctionnelles (QoS).

• ***La sélection :*** consiste à sélectionner un ensemble de services candidats pour chaque service abstrait. Les services concrets sélectionnés doivent satisfaire les contraintes globales de QoS. Pour ce faire, les auteurs introduisent une heuristique basée sur la technique de clustering, qui permet de classer les services en niveaux de qualité, utilisés pour déterminer l'utilité des services.

L'heuristique adopté, traite le problème de sélection de services en deux étapes :

1. phase de classification locale : vise à déterminer les services concrets en utilisant le clustering ; cette phase est réalisée pour chaque service abstrait du plan de composition.
2. phase de sélection globale : utilise les services obtenus de la classification locale pour orienter le choix de la composition quasi-optimale.

• ***L'exécution :*** Une fois la sélection globale est effectuée, la phase de l'exécution utilise les services sélectionnés, afin de fournir une composition de service exécutable en remplaçant chaque service abstrait dans du plan par un un service concret. Lors de l'exécution, un service est sélectionné parmi les services d'une mem classe en choisissons le meilleur en terme de QoS.

Une entité de surveillance contrôle la dégradation de la QoS pour permettre une évaluation dynamique de services afin de détecter s'il y a des services qui se diminuent en termes de QoS.

2.5.1.2 Les paramètres de QoS

Les auteures divisent les attributs de qualité de services en deux catégories :

- ***les attributs quantitatifs :*** cette categorie englobe : le temps de réponse, la disponibilité, la fiabilité et débit. Elle est divisée en deux classes : les attributs négatifs (temps de réponse et prix) et les attributs positifs (disponibilité, fiabilité et débit).
- ***Les attributs qualitatifs :*** tels que la sécurité et la confidentialité.

2.5.1.3 Algorithme de sélection de services

Pour sélectionner les meilleurs services concrets (en termes de QoS) de chaque service abstrait et répondre aux contraintes de QoS globales imposées, les auteurs proposent une

heuristique qui combine les techniques de sélection de services locales et globales. Elle comprend les étapes suivantes :

- ***Phase de mise à l'échelle :*** est une phase de prétraitement visant à normaliser les valeurs de QoS associées à des attributs positifs et négatifs en les transformant en une valeur comprise entre 0 et 1.
- ***Classification locale :*** elle vise à classer les services concrets de chaque service abstrait en fonction de leurs QoS. En utilisant une techniques de clustering, le but de cette classification est d'avoir plusieurs niveaux de service où chaque niveau contient les services ayant des valeurs de QoS proches
- ***Sélection globale :*** elle consiste à utiliser les services obtenus de la phase précedente pour sélectionner les compositions quasi-optimales.

2.5.1.4 Bilan et discussion

N. Ben Mabrouk et.al. ont proposé dans [34] une heuristique utilisant 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 des clusters. La fonction d'utilité dépend des valeurs de paramètres de qualité d'un service donné, et du nombre de services dans le cluster auquel le service en question appartient.

Les contraintes globales de QoS sont respectées lors de la composition, mais ces contraintes et le contexte de l'utilisateur peuvent être changés ou modifiés lors de l'exécution, cella n'as pas été pris en compte dans cette solution.

2.5.2 Towards composition as a service – A quality of service driven approach

2.5.2.1 Aperçu de l'apprcohe

Dans [35], les auteurs ont proposé une approche de composition CaaS (Composition as a Service), semi-automatique, combinée avec un langage spécifique pour le domaine nommé VCL (Vienna Composition Language), qui détermine hiérarchiquement les contraintes de composition.

2.5.2.2 Synthèse de CaaS

D'un point de vue de l'utilisateur final, le développeur utilise "client library", qui compile les spécifications de VCL et vérifie les erreurs statiques pour éviter une composition avec des entrées invalides. Les attributs de qualité de service qui figurent dans le tableau 2.1, ont été pris en considération dans cette approche.

Attributs	**Formules**	**Unité**
Temps de réponse	$Q_{rt}(n)=1/n\sum_{i=0}^{n} q_{rt}(i)$	msec
Latence	$Q_{la}(n)=1/n\sum_{i=0}^{n} q_{la}(i)$	msec
Disponibilité	$Q_{av}(t_0, t_1, t_d)=1-\frac{t_d}{t_1-t_0}$	pourcent
Exactitude	$Q_{ac}(r_f, r_t)=1-\frac{r_f}{r_t-}$	pourcent
Débit	$Q_{tp}(t_0, t_1, r)=1-\frac{r}{t_1-t_0}$	invocation/sec
Prix	$\frac{n}{a}$	Par invocation
Messagerie fiable	$\frac{n}{a}$	vrai, faux
Sécurité	$\frac{n}{a}$	{aucun, X.509...}

TABLE 2.1 – les attributs de QoS utilisés dans l'approche Rosenberg et al.

VCL permet de détecter quels sont les paramètres de QoS requis (paramètres globaux et locaux) dans la composition de services, sachant que les paramètres locaux peuvent être spécifiées par l'utilisateur.

Dans VCL, les auteurs ont utilisé quatre valeurs différentes hiérarchiques par défaut : requis, fort, moyen et faible. Ces valeurs peuvent être spécifiées pour chaque contrainte de QoS (globale et locale) et considérées par l'algorithme de composition.

2.5.2.3 Génération d'un service composite

La génération d'un service composite se fait en cinq étapes :

- ***Résolution d'entité :*** la résolution d'entité est le processus d'interroger tous les services candidats, qui mettent en œuvre une fonction donnée. Dans cette étape, les auteurs ont supposé que chaque fonction du domaine est définie dans le modèle métadonnées de VRESCo (Vienna Runtime Environment for Service-oriented Computing).

- ***Génération d'une Structure de Composition :*** cela se produit en parallèle avec l'étape précédente (résolution d'entité). L'objectif principal de cette étape

est l'analyse des dépendances de données pour déterminer l'ordre d'exécution correct.

- ***Contrainte de résolution et d'optimisation QoS :*** cette étape consiste à formuler un problème de satisfaction de contraintes (CSP). Les sorties de l'étape de résolution d'entité sont utilisées commme entrées dans cette étape.

- ***Génération d'une Composition exécutable :*** cette étape consiste à transformer le workflow structuré et l'information quel est le service concret à invoquer (du processus de résolution de contraintes) en une représentation Windows Workflow Foundation (WWF).

- ***Déploiement du service composite :*** il consiste à utiliser le workflow généré précédemment et l'héberger à l'aide du WWF fourni. Enfin, le point de terminaison du service composite mis en œuvre par le workflow généré est renvoyé à l'utilisateur.

2.5.2.4 Bilan et discussion

Les auteurs ont illustré le concept de «CaaS», visant à réduire le besoin d'une infrastructure de composition et permettant de composer et déployer des services à la volée. En outre, une reconfiguration dynamique d'un service composite est envisagée, si des contraintes de qualité de services (locaux et globaux), spécifiés au moment de la conception, ne sont plus respectées lors de l'exécution (basé sur le changement de la qualité de service qui implémente une fonction spécifique).

2.5.3 Combining global optimization with local selection for efficient QoS-aware service composition

2.5.3.1 Aperçu de l'apprcohe

L'existence de nombreux services web qui offrent les mêmes fonctionnalités, mais diffèrent dans les paramètres non-fonctionnels, rend la composition de services un problème de décision sur le choix des service composants en terme des éxigences fonctionnelles et QoS. Dans ce contexte, les auteurs [36] ont proposé de réaliser la composition en deux étapes, afin de sélectionner un service concret dont la valeur de QoS satisfait les exigences du client pour chaque classe des services abstraits (ensemble de services concrets).

L'environnement des services web, noté S, est défini comme l'union des classes des services abstraits S_j tel que $S_j \epsilon$ S. Un service web peut joindre et quitter une classe de

service à tout moment.

• ***Service composite abstrait :*** c'est une représentation abstraite d'une demande de composition, $CS_{abstrait} = \{S_1, ..., S_n\}$, tel que $CS_{abstraits}$ se réfère aux classes de service requis.

• ***Service composite concret :*** noté CS, c'est une instanciation d'un service composite abstrait, qui est obtenue en remplaçant chaque classe de service abstrait dans $\mathrm{CS}_{abstrait}$ par un service concret s_j tel que $\mathrm{s}_j \epsilon\ \mathrm{S}_j$.

Le vecteur Q_s, représente les attributs de QoS d'un service concret s. Les valeurs de ces attributs peuvent être collectées directement à partir des fournisseurs de services (ex. le prix), ou à partir des exécutions précédentes (ex. temps de réponse), ou encore à partir de l'expérience des utilisateurs (ex. réputation).

• ***Fonction d'utilité :*** La fonction d'utilité fait correspondre au vecteur de qualité $Q_s = \{Q_1(\mathrm{s}),\ Q_2(\mathrm{s}),\ .....,\ Q_n(\mathrm{s})\}$ une valeur réelle unique, afin de trier et classer les services candidats. Elle est utilisée ainsi pour évaluer la qualité multidimensionnelle d'un service donné.

Formellement, les valeurs maximales et minimales agrégées du k-ème attribut de QoS d'un service composite sont calculées comme suit :

$Q_{min'}(\mathrm{k}) = \sum_{j=0}^{n} Q_{min}(j,k)$ et $Q_{max'}(\mathrm{k}) = \sum_{j=0}^{n} Q_{max}(j,k)$

Tel que $\mathrm{Q}min(\mathrm{j,k}) = \min_{\forall s_{j,i} \epsilon S_j}\ \mathrm{q}_k(\mathrm{S}_{j,i})$ et $\mathrm{Q}max(\mathrm{j,k}) = \max_{\forall s_{j,i} \epsilon S_j}\ \mathrm{q}_k(\mathrm{S}_{j,i})$

Maintenant la fonction d'utilité d'un service concret $s \epsilon S_j$ est calculée comme suit :

$$U(s) = \sum_{k=1}^{r} \frac{Q_{max}(j,k) - q_k(s)}{Q_{max}(j,k) - Q_{min}(j,k)} * w_k \tag{2.1}$$

L'utilité globale d'un service composite est calculée comme suit :

$$U'(s) = \sum_{k=1}^{r} \frac{Q_{max'}(k) - q'_k(CS)}{Q_{max'}(k) - Q_{min'}(k)} * w_k \tag{2.2}$$

Tel que : $w_k \epsilon R_0^+$ et $\sum_{k=1}^{r} w_k = 1$ étant le poids de q'_k pour représenter les priorités de l'utilisateur.

2.5.3.2 Les étapes de l'approche proposée

- ***La première étape :*** elle consiste à décomposer d'une manière optimale les contraintes globale de la qualité de service C‘ en "n" contraintes locales $c_1, \ldots, c_n$, en utilisant MIP (Mixed Integer Language). puis, Chaque valeur de l'attribut de QoS est transformée en une valeur comprise entre 0 et 1.
  La plage de qualité de chaque attribut de QoS est divisée en un ensemble de valeurs discrètes de qualité, appelées des niveaux de qualité. Les niveaux de qualité sont déterminés pour représenter l'ensemble d'informations de chaque classe de services. Une valeur q_{jk}^z comprise entre 0 et 1, est affectée ensuite à chaque niveau de qualité q_{jk}^z. Cette valeur estime l'avantage d'utiliser ce dernier comme une contrainte locale.

- ***La deuxième étape :*** le compositeur de service lors envoie les contraintes locales obtenues dans la première étape, et les préférences de l'utilisateur à tous les services brokers, chacun de ces dernier effectue une sélection locale distribuée et renvoie le meilleur service web candidat au compositeur de service. La figure 2.5 illustre les deux étapes de l'approche.

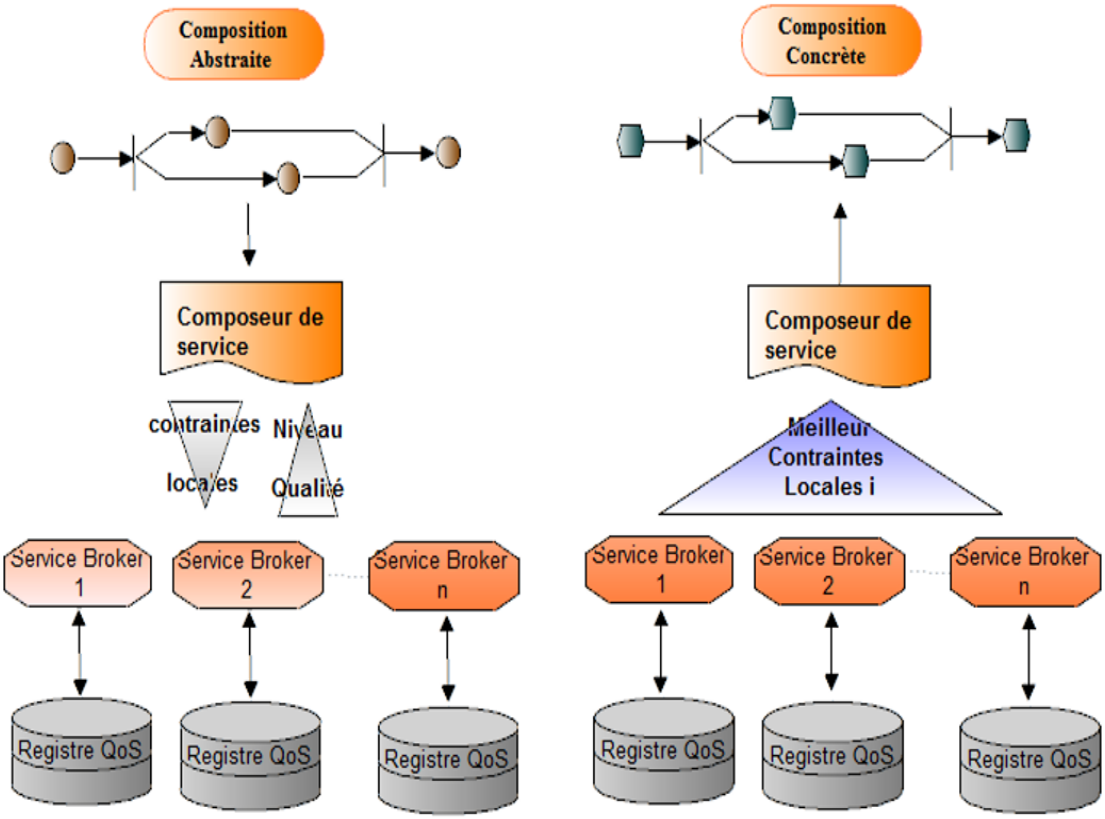


FIGURE 2.6 – Les étapes de l'approche proposée par Alrifai et Risse.

2.5.3.3 Bilan et discussion

Dans ce travail [36], les auteurs ont combiné l'optimisation globale et la sélection locale pour bénéficier des avantages des deux méthodes et se rapprocher de la composition optimale, comme il n'ont considéré que le modèle séquentiel de composition en argument, ce qui fait que les modèles non-séquentiels doivent se transformer en un modèle séquentiel. L'approche proposé n'est pas auto-adaptative car le nombre de niveaux de qualité n'est pas définit lors de l'exécution mais il est définit à priori d'une façon manuelle.

2.5.4 Context-aware Dynamic Service Composition in Ubiquitous Environment

2.5.4.1 Aperçu de l'approche

Les auteurs dans [37] proposent une approche de composition de services en couches, flexible et tolérante aux défaillances, il s'agit de générer un plan de composition, en utilisant des techniques basées sur des règles afin d'adapter la composition de services aux

changements qui se produisent sur les services et le contexte d'utilisation.

2.5.4.2 Génération de plan globale

Il conciste en la génération automatique d'un plan de composition abstrait en utilisant une technique à base de règles. Le plan contient tous les services abstraits possibles qui peuvent participer à la composition du services requis.

Initialement, le graphe G = (V, E) = (RP_0,$\emptyset$). Pour chaque paramètre de la requête demandé RP, les services abstraits sont choisis du répertoire de services abstraits (ASD : Abstracts Services Directory). Ces services abstraits sélectionnés, appelés services abstraits candidats, ont au moins un paramètre de sortie appartenant à RP tel que :

- RP_i : est le i^{ieme} ensemble des paramètres requis sachant que l'ensemble initial RP_0 représente les paramètres de sortie nécessaires de la requête T.

- Un arc de RP_i à RP_j noté (RP_i, RP_j, AS_i, R_i) : représente le service abstrait qui fournit tous ou une partie des paramètres requis RP_i et qui requiert le paramètre RP_j. Le service abstrait AS_i a une reputation RP_i.

Si toutes les données fournies par les services abstraits candidats incluant les paramètres RP_i demandés, tous les paramètres d'entrée des services en question sont ajoutés à V et tous les arrêtes correspondantes (RP_i, RP_k, AS_i, R_k) sont ajoutés à E, sinon, le RP_i est supprimé du graphe car aucune solution n'existe. La génération du graphe G s'arrête lorsque tous les paramètres demandés sont obtenus ou dans le cas opposé, la composition est impossible.

2.5.4.3 Génération du plan optimal

Cette couche utilise le plan global obtenu à l'issue de la première couche d'une part pour la production d'un plan abstrait optimal de composition, et d'autre part, pour la regénération d'un nouveau plan optimal en cas d'échec.

Une fois le plan global est obtenu, un plan abstrait de composition optimal est choisi en fonction de la réputation et de la complémentarité des paramètres fournis par les services candidats. Cette sélection permet d'éviter l'invocation des services qui fournissent les mêmes paramètres. Par conséquent, le plan optimal généré a le plus petit nombre de

services et de paramètres peuvant apparaître dans la composition.

Une sélection automatique est effectuée pour éviter la phase de redécouverte en cas d'échec en sélectionnant localement le nouveau plan optimal. La génération du plan optimal repose sur des règles et des techniques de classement telles que les services abstraits sont classés selon la réputation et la complémentarité de leurs paramètres de sortie.

• **Sélection à base de classement de services**

Après la génération du plan optimal abstrait, un service concret qui possède une QoE (qualité de l'expérience de l'utilisateur) supérieur à un seuil spécifié par l'utilisateur est sélectionné pour chaque service abstrait

• **Calcule de la QoS et estimation de QoE**

Le processeur de contexte associe un poids à chaque paramètre de QoS en fonction des préférences de l'utilisateur, du contexte de l'utilisateur et de l'environnement.

• **Estimation de la réputation d'un service abstrait**

La réputation d'un service abstrait dépend de la qualité de ces services concrets, plus précisément, elle dépend du succès ou de l'échec de l'execution du service concret, la formule de calcul de l'estimation est donnée comme suit :

$$R_t(a) = \frac{Nb_{success}}{Nb_{selectio}} \tag{2.3}$$

Où (R_t) représente la réputation du service abstrait un à l'instant t.

Si la sélection du service abstrait A est réussie, sa réputation augmente. Sinon, si le service abstrait ne trouve pas un service concret valide ou le service concret choisi n'est pas réussi, alors sa réputation diminue.

• **Contrôle du plan d'exécution**

En raison de la mobilité des utilisateurs et des dispositifs, le comportement non déterministe des services, et la nature incertaine des environnements omniprésents, le plan de composition doit être souple et tolérant aux pannes. A cet effet, l'approche proposée intègre un mécanisme d'apprentissage sur la QoS du service concret et de la réputation du

service abstrait. Tous les paramètres de qualité de tous les services concrets et la réputation de tout service abstrait sont également initialisés.
Selon le résultat du service invoqué, (succès ou échec) le mécanisme d'apprentissage met à jour le service concret en calculant ses nouveaux paramètres de qualité et estime la nouvelle réputation de son service abstrait.

2.5.4.4 Bilan et discussion

K. Tari et al. [37] proposent une approche en couche permettant d'optimiser le nombre de services concrets, les paramètres inclus dans la composition et le temps d'exécusion dans les environnements à grande échelle en supprimant la phase de redécouverte. Dans cette approche, le contexte n'a pas été pris en compte dans le processus de sélection et de composition de services. De plus la QoS n'a pas été clairement développée, c'est à dire, qu'aucune description formelle n'a été proposée pour évaluer la QoS du service concret. L'inconvénient de cette approche est l'implication des utilisateurs pour assurer le système de réputation, ce qui remet en cause la sélection automatique et autrement basée sur la confiance entre les utilisateurs.

2.5.5 Flexible service selection with user-specific QoS support in service-oriented architecture

2.5.5.1 Aperçu de l'apprcohe

Dans cet article [38], les auteurs proposent un modèle de sélection pour les systèmes basés sur SOA (Service Oriented Architecture), l'algorithme recommande le nombre de services condidats basé sur les contrainte de QoS exigées par l'utilisateur.

L'algorithme [38] est nommé SPSE "Service Providers Search Engine", il ne prend pas de décision de la planification pour une tâche, mais c'est un outil assistant de la sélection de service. Il est caractérisé par :

- sa capacité de gérer un grand nombre de services et d'utilisateurs.
- Il peut être employé par différents utilisateurs ayant des préférences différentes sur les services.
- Il supporte plusieurs paramètres de qualités y compris le temps, le prix, le degré de confiance, etc. Deplus, il peut être étendu pour supporter plus de critères de QoS.
- Il reçoit les valeurs exactes des préférences des utilisateurs automatiquement.

2.5.5.2 Les paramètres de qualité de service

• ***Degré de confiance :*** implique le niveau de fiabilité d'un service. Le degré de chaque fournisseur de service, est calculé en agrégeant plusieurs facteurs (taux de réussite, l'évaluation des utilisateurs pour le service, niveau de fiabilité) par la formule suivante :

$$Degre = \sum_{i=1}^{d} (w_i * u_i) \tag{2.4}$$

Tel que w_i est le poids du facteur correspondant et d est le nombre de facteur.

• ***Temps de réponse :*** représente le temps nécessaire pour obtenir un résultat, il peut être estimé en utilisant les techniques de performances existantes (ex. historique de service, modèle analytique, etc.).

• ***Prix :*** représente le coût monétaire d'un service obtenu en compensant le coût du matériel et des logiciels.

2.5.5.3 Les étapes du modèle proposé

a) Emetteur de tâche : il se charge d'accepter la requête, il formule et sauvegarde les informations d'entrée. Après la détermination du fournisseur de service par le scheduler (ordonnanceur), le composant de l'émetteur de tâche envoie les informations aux fournisseurs.

b) Scheduler : il communique avec le registre UDDI, pour déterminer le fournisseur de service le plus approprié à une requête donnée. Il consiste en examinateur, sélectionneur, et plan d'exécution.

- *Demandeur :* il communique avec le registre UDDI pour trouver tous les fournisseurs pouvant prendre en charge la requête de l'utilisateur, et calcule les valeurs de QoS.
- *Sélectionneur :* il retourne la liste des fournisseurs de services candidats, l'algorithme SPSE est développé dans le sélectionneur.
- *Plan d'exécution :* reçoit la décision finale de l'utilisateur, et met à jour ses préférences.

c) Moniteur d'exécution : il se charge de la surveillance de l'état d'exécution de la requête. Si le fournisseur de service n'est plus disponible, il active le scheduler pour trouver un autre fournisseur. À la fin de l'exécution, il se charge de collecter les résultats de la

requête.

Généralement, programmer un workflow pour un SOA demande trois étapes : demande de programmer une tâche, programmation de la tâche et représentation de la tâche. Le noyau de ce travail, est de programmer chaque tâche avec une convention de QoS. Après la génération du plan de sortie, le workflow sera présenté au système. Les services qui satisfont les QoS seront sélectionnés.

2.5.5.4 L'algorithme de planification de service SPSE

Le modèle de service de tous les fournisseurs doit être sous la forme d'un modèle Job. L'algorithme SPSE consiste à fournir une liste de tous les fournisseurs de servcices, qui sera filtrée en supprimant les services qui ne respecte pas le théorème "minimiser le coût et maximiser le degré de confiance", après avoir trier la liste obtenue, de telle sorte que le meilleur fournisseur a le rang minimal, l'utilisateur sélectionne un service. Enfin, les préférences de l'utilisateur sont mises à jour.

2.5.5.5 Bilan et discussion

Dans ce travail [38], les auteurs ont proposé un algorithme flexible qui satisfait les multiples exigences de QoS de l'utilisateur, et supporte ses préférences. En plus, le mécanisme de SPSE adapté pour les systèmes à grande échelle. L'algorithme SPSE donne à l'utilisateur la possibilité de sélectionner un service ce qui rend l'approche semi-automatique. si l'utilisateur n'est pas conscient de son environnement c'est possible qu'il ne réussit pas à choisire le bon service.

2.5.6 Composite service adaptation : a QoS-driven approach

2.5.6.1 Aperçu de l'apprcohe

Cette approche [39] est inspiré des fourmis, où les fourmis coopèrent pour trouver les meilleurs chemins vers les meilleures sources de nourriture. Elle porte sur un système multi-agent dans un environnement dynamique, où chaque service composant peut être représenté par un agent.

2.5.6.2 Techniques utilisées

• **Optimisation de colonie de fourmis**

La sélection des services est basée sur le concept de l'ACO (Ant Colony Optimization) où les fourmis coopèrent pour trouver les meilleurs chemins vers les meilleures sources de nourriture, Les fourmis du même nid doivent trouver les meilleures sources de nourriture. La qualité d'une source de nourriture est définie par la distance au nid et de l'énergie qui peut être obtenue à partir d'un type spécifique de nourriture. Les fourmis doivent avoir une forme d'évaluation, afin de déterminer la meilleure source de nourriture disponible. Pour résoudre des problèmes dans des environnements dynamiques, la fourmi ajoute et supprime des chemins à la source de nourriture si une source de nourriture est bonne ou pas.

Dans les algorithmes ACO pour organiser la diffusion des agents d'information dans l'environnement, le problème est modélisé sous forme d'un graphe, où les agents recherchent les meilleurs chemins. Les politiques mises en œuvre par les agents, utilisent seulement de l'information locale disponible dans l'environnement et l'état interne des agents. Cette information est créée par chaque agent associé à un service.

Quand une fourmi trouve une solution (un chemin), elle diffuse dans l'environnement une information, appelée phéromone artificiel. D'autres fourmis peuvent ensuite utiliser cette information pour leurs propres solutions.

• **La recherche de nourriture (services)**

Les auteurs modélisent deux types d'entités :

✓ ***OrgAgents :*** représentent le service composite dans la superposition de services, ils sont chargés de donner le meilleur ensemble de services pour leur service composite. OrgAgents sont responsables de sélectionner les services qui participeront à une composition.

✓ ***RessourceAgent :*** chaque RessourceAgent est associé à un service, qui est responsable du stockage des informations de QoS du service qu'il représente, il possède des références à d'autres ResourceAgents dans l'environnement. Et deux types de fourmis : ***ExplorationAnts*** *;* ***IntentionAnts****.*

OrgAgents envoie un certain nombre de *ExplorationAnts* à la recherche de services dans

l'environnement, les *ExplorationAnts* vérifient la QoS des services composants déposent cette information dans les *ResourceAgents*, et évaluent la QoS agrégée en utilisant une heuristique définie par l'*OrgAgent*. Chaque *ExplorationAnt* a un temps de survie, Si l'*ExplorationAnt* n'est pas capable de trouver une solution, l'*ExplorationAnt* s'arrête. Les *ExplorationAnts* renvoient les informations recueillies à l'*OrgAgent*. Ce dernier, à son tour, évalue les solutions apportées par les *ExplorationAnts* et sélection une des solutions, une fois que l'*OrgAgent* décide de s'engager dans un trajet de composition particulier, il envoie *IntentionAnts* à tourner le long de ce chemin de compostion.

2.5.6.3 Bilan et discussion

La sélection des service dans [39], est basée sur les concepts de l'algorithme ACO où un problème de composition de service est modélisé sous forme d'un graphe dans lequel, chaque service est associé à un agent. Il [39] créent un réseaux, fondé sur les relations entre services puis créer un graphe virtuelle, où les fourmis virtuelles (agents) font leurs recherches, à base d'une configuration donnée par le développeur de l'application, qui est essentiellement l'ensemble des contrats entre l'application de l'utilisateur et les services participant dans les compositions des services.

2.5.7 Towards an event-aware approach for ubiquitous computing based on automatic service composition and selection

2.5.7.1 Aperçu de l'apprcohe

Dans [40], les auteurs adoptent une approche à base de règles, pour générer automatiquement deux types de graphes de composition de services en deux phases principales : phase hors ligne et une phase en ligne. Dans la phase hors ligne, un graphe global qui relie tous les services abstraits disponibles est construit. Dans la phase en ligne, un sous-graphe qui répond à l'objectif spécifié est extrait à partir du graphe global.

2.5.7.2 Modèle de composition de services

La composition de service fait le raisonnement sur les entrées et les sorties des services abstraits ; Il s'appuie sur le rejet des services inutiles pour participer au plan de composition.

2.5.7.3 Définition des règles de composition de services

• ***Règle de marquage :*** elle élimine la confusion sur les sources de tous les paramètres qui participent à la composition de service et réduit leur nombre total en ne conservant

que les services pour lesquels au moins un paramètre de sortie est utilisé dans le graphe de composition.

• ***Règle d'égalité :*** Lorsque deux services apportent les mêmes valeurs ajoutées à la composition, on définit la règle d'égalité pour supprimer le plus grand service en termes de nombre de paramètres d'entrée. Ce choix est motivé par le fait qu'un service qui possède plusieurs paramètres d'entrée sera probablement relié à plusieurs autres services dans la composition.

• ***Règle de simple inclusion :*** Le service dont les sorties sont incluses dans les résultats d'un autre service devrait être supprimé du graphe de composition car il n'apporte pas de valeur ajoutée à cette composition.

• ***Règle de d'inclusion complexe :*** Un service sera supprimé du graphe de composition si un de ses paramètres de sortie sont dans les sorties d'une combinaison d'autres services qui sont satisfaites au même niveau du graphe de composition.

• ***Règle de démarquage :*** Elle consiste à supprimer de la table de marquage tous les services abstraits qui ne participent pas à la réalisation d'un objectif donné.

2.5.7.4 Construction du graphe Globale

Un graphe abstrait de la composition de services (AGoSC) est généré en appliquant un algorithme pour relier les services abstraits disponibles en utilisant de manière appropriée les quatre règles définies précédemment. L'algorithme proposé est un planificateur représenté par un couple (AS, satisfiedAbstractServices) tels que AS est un ensemble de tous les services abstraits disponibles et *satisfiedAbstractServices* contient tous les services abstraits satisfaits au niveau du graphe AGoSC.

Initialement, tous les services abstraits conçus pour l'acquisition du contexte sont considérés comme satisfaits. Par la suite, un niveau abstrait (noté *abstractGraphLevel*) est réalisé à chaque itération de l'algorithme, chaque service abstrait est mis à jour en supprimant tous les services qui sont déjà vérifiés et satisfaits afin d'éviter les répétitions lors de la prochaine vérification.

2.5.7.5 Extraction de sous-graphe

Un graphe de composition de services (SubGoSC) est extrait à partir du graphe global AGoSC en utilisant la règle de démarquage et l'objectif fixé G. C'est un graphe partiel qui peut répondre à la règle de l'événement détecté. L'algorithme commence à partir du dernièr niveau du graphe abstrait AGoSC jusqu'au premier niveau, en explorant les niveaux intermédiaires. A chaque niveau, la règle de démarquage est appliquée, afin de sélectionner les services qui assurent au moins un paramètre de l'objectif fixé. Cette méthode est basée sur la technique guidée par un objectif. Un sous-graphe sera donc extrait en trouvant tous les prédécesseurs de services qui offrent un ou plusieurs paramètres de l'objectif initial. Concrètement l'objectif initial est spécifié dans la règle de l'événement et l'extraction de sous-graphe est effectuée lorsqu'un événement est détecté. Donc, SubGoSC est alors considéré comme un graphe partiel qui peut répondre à la règle de l'événement détecté.

2.5.7.6 Bilan et discussion

dans l'approche proposée [40], la communication entre les différents services est établie et gérée par le système ; ce qui diminue les performances du système en termes d'exécution des compositions de services. Les événement détecté sont traités d'une façon individuelle ; Toutefois, des corrélations peuvent exister entre les événements qui se produisent dans un environnement ambiant, donc, il est nécessaire d'augmenter le système par des modèles de corrélation entre événements afind'améliorer l'analyse et l'interprétation des situations et encore réduire le nombre d'événement à traiter.

2.5.8 A context-aware computing mediated dynamic service composition and reconfiguration for ubiquitous environment

2.5.8.1 Aperçu de l'apprcohe

Afin de prendre en charge la sensibilité au contexte, les auteurs de [41], ont proposé une architecture SOA étendue qui intègre un système de prise de décisions sensible au contexte permettant d'évaluer les services pour une composition dynamique de services et une reconfiguration selon les informations de contexte.

2.5.8.2 L'architecture globale

L'architecture proposée dans de l'approche [41] est divisée en quatre parties :

• ***Environnement de l'utilisateur :*** un composant de génération d'un BPEL abstrait, est capable de composer de multiples services web abstraits (AWS). Un AWS est une catégorie de services ayant des fonctions et des interfaces équivalentes.

• ***Environnement de service ubiquitaire :*** tous les dispositifs et les applications web partagent leurs données et capacités via des interfaces standars de services web.

• ***Business Process Agent (BPA) :*** Les services web fournis par le middleware de service broker (SBM) sont des services web virtuels (VWS). Un VWS est un service broker qui transmet dynamiquement des messages SOAP (Simple Object Access Protocol) à un service web concret et renvoie le résultat au moteur BPEL, afin qu'il puisse sélectionner dynamiquement un service à invoquer parmi les services candidats selon le contexte. Le composant de génération et de déploiement de BPEL, peut convertir un BPEL abstrait en BPEL, en remplaçant AWS dans le BPEL abstrait par le VWS en fonction du contexte.

• ***Système de décision sensible au contexte (CDMS) :*** le composant d'évaluation de services évalue automatiquement les services web en fonction du contexte, et les scores marqués sont basés sur la QoS et les préférences de l'utilisateur. Le composant d'interprétation du contexte utilise un modèle sémantique multi-dimensionnel.

2.5.8.3 Web Service abstrait (AWS)

AWS est une description ontologique de services réels. Les AWSs sont classés hiérarchiquement. Les fonctionnalités d'une couche supérieure (resp. inférieure) d'un AWS sont plus abstraites (resp. concrète) que celles d'une couche inférieure (resp. supérieure) d'un AWS. Dans cet article [41], les auteurs ont proposé un modèle multi-dimensionnel sémantique de l'information de contexte : qui (Who), où (Where), quand (When) et quoi (What). La dimension who représente les entités qui effectuent certaines actions, la dimension where représente les lieux dans le monde physique, la dimension when indique quand l'activité a eu lieu et la dimension what décrit les activités réalisées dans le monde physique.

2.5.8.4 Evaluation des services

Le système proposé évalue les services en fonction de la QoS et des préférences de l'utilisateur. Les auteurs ont utilisé la formule suivante pour calculer le score de QoS d'un service :

$$P_{QoS} = \sum_{i=1}^{m} (C_i * P_i) \tag{2.5}$$

Tel que m est le nombre de d'attribut de QoS d'un service.
P_i est Le score d'un facteur impact.
C_i est le poids de P_i, il indique le degré d'influence d'un facteur impact et ($0 \leq C_i \leq 1$, $\sum_{i=1}^{m} C_i = 1$).

La fréquence d'utilisation d'un service reflète les préférences de l'utilisateur pour ce service. Dans ce travail [41], des règles d'association ont été utilisées pour obtenir les associations entre les services et les dimensions des événements ou des scénarios.

Quand un nouvel événement ou scénario se produit, le système proposé découvre tous les VWSs et les services concrets qui sont les plus appropriés pour l'environnement courant. Le système collectera ces services pour générer des configurations pour tous les processus métiers (BPA). Une configuration des BPA est le résultat de sensibilité au contexte. Il peut guider le processus métier pour s'adapter aux changements de l'environnement.

2.5.8.5 Bilan et discussion

Le développement des systèmes sensibles au contexte est généralement une tâche complexe qui demande beaucoup de travail et de patience par les développeurs. Comme les environnements d'intelligence ambiante est en constante évolution, les services doivent être sensibles au contexte de l'environnement et de l'utilisateur, qui sont considérés comme indisponsable pour l'interaction entre un utilisateur et une application. Pour cela, les auteurs ont donné une grande importance au contexte et à la QoS en utilisant la technique de workflow pour une composition dynamique.

Un langage de programmation est utilisé et destiné à l'exécution des procédures d'entreprise AbstractBPEL (Abstract Business Process Execution Language) dans SOA, qui est dérivé de XML. Ce langage peut migrer entre différents environnements et peut être changé dynamiquement pour s'adapter aux changements dramatiques de l'environnement.

2.5.9 Flexible QoS-Aware Service Composition in Highly Heterogeneous and Dynamic Service-Based Systems

2.5.9.1 Aperçu de l'apprcohe

SOC (Service Oriented Computing) permet la création des applications complexes, en composant des services, qui fournissent des fonctionnalités qu'aucun service parmi les composants n'est capable de les fournir seul.

Dans cet article [42], les auteurs ont proposé un moyen flexible de formuler des configurations de composition appropriées pour les systèmes à base de services et ils offrent des modèles de mobilité décrivant comment les pompiers (FF_s) se déplacent en se basant sur une organisation hiérarchique (en niveau). Deux concepts ont été utilisés : Service Concret qui se réfère à un service invocable, et Service Abstrait qui définit de manière abstraite les fonctionnalités d'un service concret. Une application peut être considérée comme une composition de services abstraits et elle peut être représentée sous forme d'un workflow.

Dans cette approche [42], la coordination de l'application est distribuée sur de nombreux nœuds. Chaque nœud orchestré intègre un moteur de workflow locale et n'a qu'une vue partielle de la composition globale. Ces nœuds coopèrent les uns avec les autres dans la réalisation de la demande complète.

Les attributs suivants ont été considérés pour évaluer la qualité d'une configuration de l'application composite :

- ***Le temps de réponse (QR_{RT}) :*** est l'intervalle de temps entre le moment de l'envoie de la requête et le moment où la réponse est reçue.
- ***La consommation de la batterie (QBC) :*** est la différence de l'énergie observée dans les nœuds participant à réaliser une composition de service.
- ***Le taux de succès (QSR) :*** représente la fraction des données échangées entre les nœuds qui collaborent avec succès dans une configuration de composition.

2.5.9.2 Vue générale de l'approche proposée

La figure 2.7 présente une vue générale du processus de composition de services dans des environnements hétérogènes hautement distribués. Tout d'abord, les utilisateurs soumettent une demande avec un ensemble d'objectifs de qualité pour une application composite dont le plan abstrait est déjà stocké dans le moteur de composition (1). Ensuite,

le sous-programme de découverte de service / ressources (2), découvre les ressources qui sont disponibles pour la réalisation de l'application demandée. Lespace de toutes les configurations possibles de réalisation est formulé (3). Enfin, un processus d'optimisation de recherche, sur cet espace de configurations possibles, sélectionne la configuration qui satisfait au mieux les objectifs demandés(4).

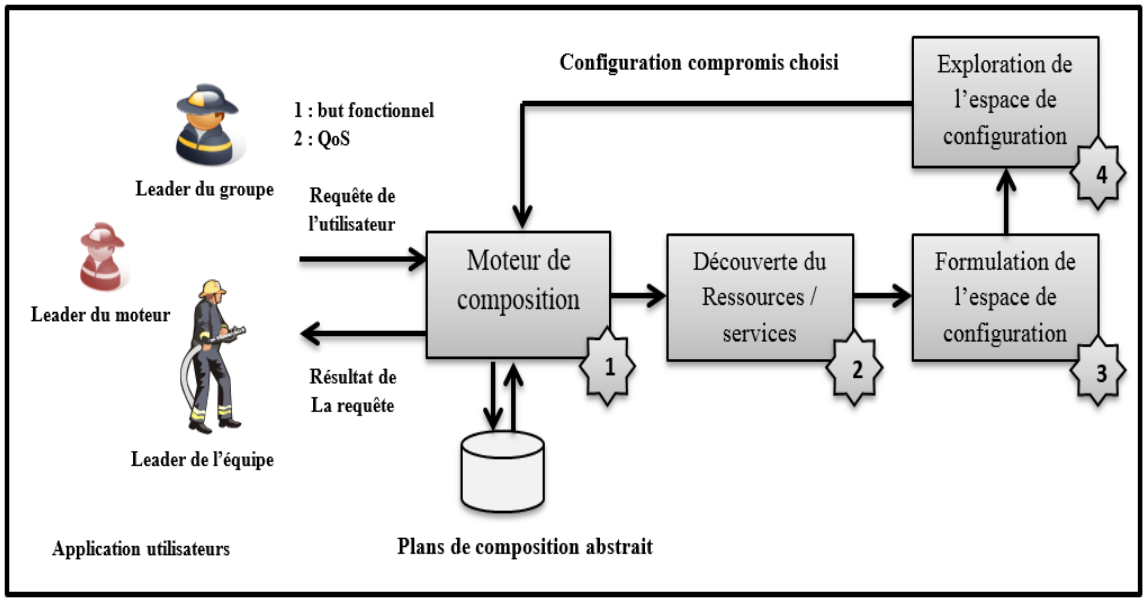


FIGURE 2.7 – Vue générale de l'approche proposée par Efstathiou et al.

2.5.9.3 Degrés de liberté

Le degré de liberté (DOF) est un paramètre de composition qui est libre d'être modifiée pour affecter la qualité de service de l'application réalisée, tout en laissant sa partie fonctionnelle non affectée. Dans ce travail [42], les types possibles de changements ont été regroupés dans une configuration de composition en trois degrés de liberté : les services concrets sont sélectionnés pour mettre en œuvre les services abstraits de la composition ; la répartition du workflow en sous-orchestrations et dans quel nœud sont déployées les sous-orchestrations sélectionnées.

2.5.9.4 Sélection de services

Pour chaque service abstrait qui décrivent une fonctionnalité nécessaire, il peut exister de nombreux services concrets qui répondent à la même fonctionnalité, éventuellement avec différentes propriétés de qualité de service. Sur la base des objectifs de qualité de service

souhaités, ainsi que les contraintes de QoS offertes par les services, ces derniers peuvent être utilisés de manière interchangeable dans une composition.

2.5.9.5 partitionnement d'orchestration

Les auteurs de [42], ont proposé de décentraliser de manière flexible l'orchestration à la demande en sous-orchestrations. En effet, le plan initial est divisé en sous-workflows, tout en respectant le flux de contrôle du plan initial. Ainsi, cette approche est capable de produire des plans avec différents niveaux de décentralisation : de l'orchestration entièrement centralisée à totalement décentralisée.

2.5.9.6 Sélection du nœud orchestrateur

Après avoir partitionné le plan initial en sous-workflows, la dernière étape consiste à attribuer les sous-workflow pour orchestrer des nœuds. Le choix d'un nœud d'orchestration peut dépendre de ses ressources disponibles, telles que le niveau de la batterie, la disponibilité des nœuds, etc.

2.5.9.7 Bilan et discussion

Dans cette approche [42], les auteurs ont présenté une formulation flexible de l'espace de configuration de composition de service, qui est adaptée à la nature hautement distribué et hétérogène de systèmes mobiles. En revanche cet approche, favorise l'idée de décentraliser de manière flexible la tâche de l'orchestration d'une application composite en sous-orchestrations, en sélectionnant les nœuds appropriés pour orchestrer les sous orchestrations résultantes.

La nature dynamique des systèmes mobiles rend la prise en considération du contexte très importante. Pour cela, les auteurs ont donné de l'importance au contexte, ainsi qu'à la QoS qui forme aussi un critère primordial pour la sélection de services.

2.6 Synthèse

La composition de services dans les environnements ubiquitaires est un thème de recherche qui est d'actualité. Malgré le nombre important de travaux réalisés sur ce sujet, la composition de services reste un problème très complexe. Cette complexité est due au fait que les solutions doivent tenir compte du nombre croissant de services, de leur mise à jour continue et de leur hétérogénéité. En conséquence, la composition de service

nécessite de traiter les problématiques de la sélection des services, de leur exécution et de leur modèles d'interaction.

Les approches étudiées dans ce chapitre, se focalisent sur un moteur de composition (orchestrateur), ce qui augmente le risque de tomber sur un point de défaillance central. Aussi la dynamicité des environnements ubiquitaires rend la prise en compte du contexte importante comme dans les approches [38, 42, 37, 40]. Contrairement à ces dernières, les propositions dans [34, 35, 36, 39] n'ont pas pris le contexte en considération. Dans [35, 36, 40] la tolérance aux défaillances a eu une grande importance contrairement à [36, 38, 35, 37, 40].

La QoS quant à elle, est adoptée dans toutes les approches [34, 35, 36, 38, 37, 40, 39], cela permet de mieux répondre aux éxigences de l'utilisateur et de prendre ses préférences en considération. Les approches de composition étudiées dans ce chapitre déjà vues se diffèrent par rapport à la technique utilisée pour la composition (basée sur les workflow, SMA, etc.), notamment le degré d'automatisation (automatique, semi-automatique, manuelle), le contexte et les attributs de qualité de service pris en compte. Le tableau 2.2 est une synthèse des approches étudiées.

Approche	QoS	Technique	Automatique	Dynamique	Contexte	Tolérence au pannes
QoS-aware Service Composition in Dynamic Service Oriented Environments [34]	Oui	planification	Semi	Oui	Non	X
Towards composition as a service-A quality of service driven approach [35]	Oui	Workflow	Semi	Oui	X	Oui
Combining global optimization with local selection for efficient QoS-aware service composition [36]	Oui	Workflow	Oui	Oui	Non	Non
Context-aware Dynamic Service Composition in Ubiquitous Environment [37]	Oui	planification	Oui	Oui	Oui	Oui
Flexible service selection with user-specific QoS support in service-oriented architecture [38]	Oui	Workflow	Semi	Dynamique	Non	Oui
Towards an event-aware approach for ubiquitous computing based on automatic service composition and selection [40]	Oui	Planification	Oui	Oui	Oui	Non
A context-aware computing mediated dynamic service composition and reconfiguration for ubiquitous environment [41]	Oui	Workflow	Non	Oui	Oui	Oui
Flexible QoS-Aware Service Composition in Highly Heterogeneous and Dynamic Service-Based Systems [42]	Oui	Hybride	Oui	Oui	oui	Non

TABLE 2.2 – Tableau de comparaison des approches étudiées

2.7 Conclusion

L'informatique ubiquitaire rend l'information accessible n'importe où et n'importe quand, cela donne la possibilité à l'utilisateur de se situer dans des endroits différents en garantissant toujours son accès à l'information (adaptation au contexte), en lui fournissant (sélection et/ou composition) des services qui répondent à ses exigences (QoS).

Dans ce chapitre, nous avons tout d'abord présenté la composition de services, puis nous avons étudié et effectué une comparaison de quelques approches existante dans la littérature, de la découverte et la composition de services. Chacune des approches étudiées traite un point particulier et utilise des techniques appropriées.

3

Composition de services avec QoS basée sur les algorithmes génétiques

3.1 Introduction

Dans un environnement ubiquitaire, un ensemble d'équipements informatiques intelligents communiquent et collaborent ensemble en percevant le contexte global et en réagissant pro-activement afin de fournir des services adaptés au contexte et répondant aux exigences complexes de l'utilisateur. L'objectif de la composition de services est de créer de nouvelles fonctionnalités en combinant plusieurs services atomiques (ou complexes) offerts par l'environnement en vue d'apporter une valeur ajoutée en fonction de la requête de l'utilisateur.

Dans ce chapitre, nous présentons notre contribution, une approche pour le problème de la composition dynamique de services avec QoS et sensible au contexte. Elle est basée sur les algorithmes génétiques dont le but est de trouver une solution (une combinaison de services) qui répond au mieux à la requête de l'utilisateur.

3.2 Motivation

Aujourd'hui le besoin d'un client est devenu de nature multi-objectif (minimiser les coûts, minimiser les délais, maximiser la sécurité, etc.). De plus, étant donné que les services ayant des fonctionnalités similaires soient fournis par des fournisseurs offrant des propriétés non fonctionnelles différentes, il sera donc nécessaire de trouver un outil d'aide à la décision et adapté au contexte afin choisir le service le plus adéquat au besoin de l'utlisateur.

De ce fait, sélectionner un service qui intéresse les usagers est une chose, alors que découvrir le service le plus adéquat en est une autre. La qualité de service (QoS) se mesure à l'aide de plusieurs métriques (exemple, disponibilité, fiabilité, etc.). La découverte de service permet certainement de trouver de nombreux services offrant les mêmes fonctionnalités et qui répondent aux exegences de la QoS. Un challenge important est : "**lequel parmi ces services sera le plus fiable, le moins cher, le plus rapide, etc.**". Autrement dit, "**lequel sera le meilleur à sélectionner ?**". Il devient ainsi nécessaire de choisir les services judicieusement parmi ceux trouvés. Les meilleurs services participeront à une composition qui fournit des fonctionnalités avec une valeur ajoutée qu'un seul service atomique ne peut fournir. En outre, il est nécessaire aussi de fixer des critères pour sélectionner la meilleure composition parmi celles éxistantes en terme de QoS.

3.3 Représentation des services

Dans ce travail, on considère les deux types de service : service concret et service abstrait.

3.3.1 Service abstrait

Un service abstrait est un ensemble de services concrets, qui ont les mêmes entrées et les mêmes sorties, sans les précisions techniques utiles lors de la sélection de services [43]. Un service abstrait est noté : $AS_i =< AS_i^{in}, AS_i^{out}, AS_i^{cs}, R_i >$ tels que (voir la figure 1) :

AS_i^{in} : ensemble des entrées du service AS_i.

AS_i^{out} : ensemble des sorties du service AS_i.

AS_i^{cs} : ensemble de services concrets du service AS_i tel que :

AS_i^{cs} : $\{cs_{i,1}, cs_{i,2}, ..., cs_{i,n}\}$ tel que $\forall cs_{i,j}$ et $cs_{i,k} \in AS_i^{cs}, (cs_{i,j}^{in} = cs_{i,k}^{in} = AS_i^{in}) \wedge (cs_{i,j}^{out} : cs_{i,k}^{out} = AS_i^{out})$.

R_i : la réputation du service abstrait.

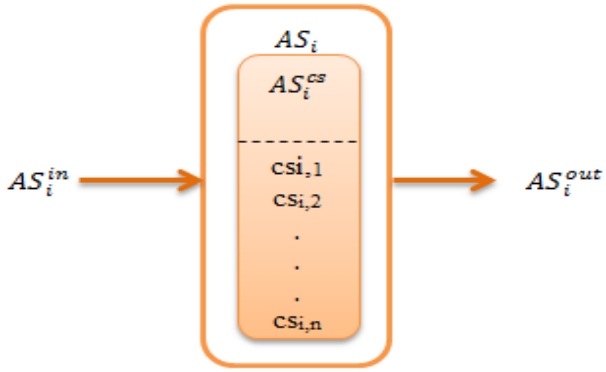


FIGURE 3.1 – Représentation d'un service abstrait.

3.3.2 Service concret

Un service concret (noté cs_i) est une instanciation du service abstrait. Il est définit comme étant une fonction physique ou logique fournissant des données contextuelles et qui peut avoir un effet sur le monde réel [7]. Il prend des données en entrée cs_i^{in}, afin de produire des données en sortie cs_i^{out}. Un service concret est subdivisé en deux parties la partie fonctionnelle et la partie non fonctionnelle. La partie fonctionnelle est alimentée par des messages d'entrée cs_i^{in} et produit des messages de sortie cs_i^{out}. La partie non fonctionnelle, quant à elle, représente la QoS du service cs_i (voir la figure 2). Un service concret est décrit par un vecteur $cs_i =< cs_i^{in}, cs_i^{out}, QoS_i >$, tel que :
cs_i^{in} : représente les paramètres d'entrée du service cs_i.
cs_i^{out} : sont les paramètres de sortie du service cs_i.
QoS : désigne les paramètres de QoS représentés sous forme d'un vecteur $QoS_i = \{q_{i,1}, q_{i,2}, ..., q_{i,n}\}$, où $q_{i,j}$ représente l'attribut j de QoS du service cs_i.

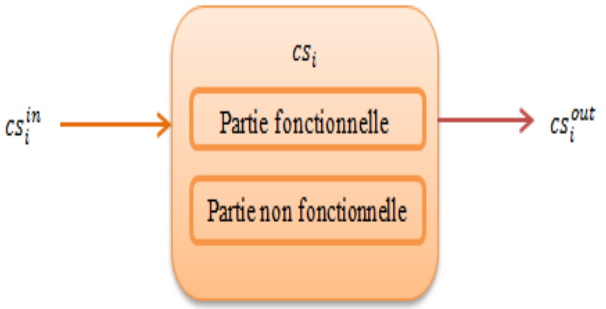


FIGURE 3.2 – Représentation d'un service concret.

3.4 Contexte

Vue la dynamicité des environnements ubiquitaires, l'adaptation au contexte est importante dans la composition de service. Les attributs contextuels sont nombreux (localisation, température, luminosité, niveau d'énergie, etc.).

Dans ce travail, on s'intéresse à deux attributs contextuels qui sont : le niveau d'énergie et la localisation.

3.4.1 Modélisation du conexte

La modélisation présentée dans cette section est inspirée des différentes définitions du contexte et des traveaux de [44]. Nous proposons un modèle de contexte où les informations contextuelles sont regroupées dans quatre catégories (utilisateur, dipositifs, services et environnement).

3.4.1.1 Contexte utilisateur

Cette classe permet de décrire les caractéristiques statiques (exemple, le nom) de l'utilisateur, les caractéristiques dynamiques (exemple, la localisation) et les préférences de l'utilisateur pour chaque attribut de QoS.

3.4.1.2 Contexte dispositif

Cette classe contient toutes les informations sur les dispositifs dont dispose l'utilisateur tels que, le terminal utilisé, le réseau de communication, les appareils qui assistent l'utilisateur dans sa vie quotidienne, etc. Ainsi que les capacités de ces dipositfs (exemple, la puissance de calcul, le niveau d'énergie, et d'autres).

3.4.1.3 Contexte services

Cette classe représente les paramètres non fonctionnels du service (QoS), et la description du service.

3.4.1.4 Contexte environnement

Cette classe regroupe les informations du contexte spécifiques à un domaine donné (exemple, information sur l'humidité, la température, etc.).

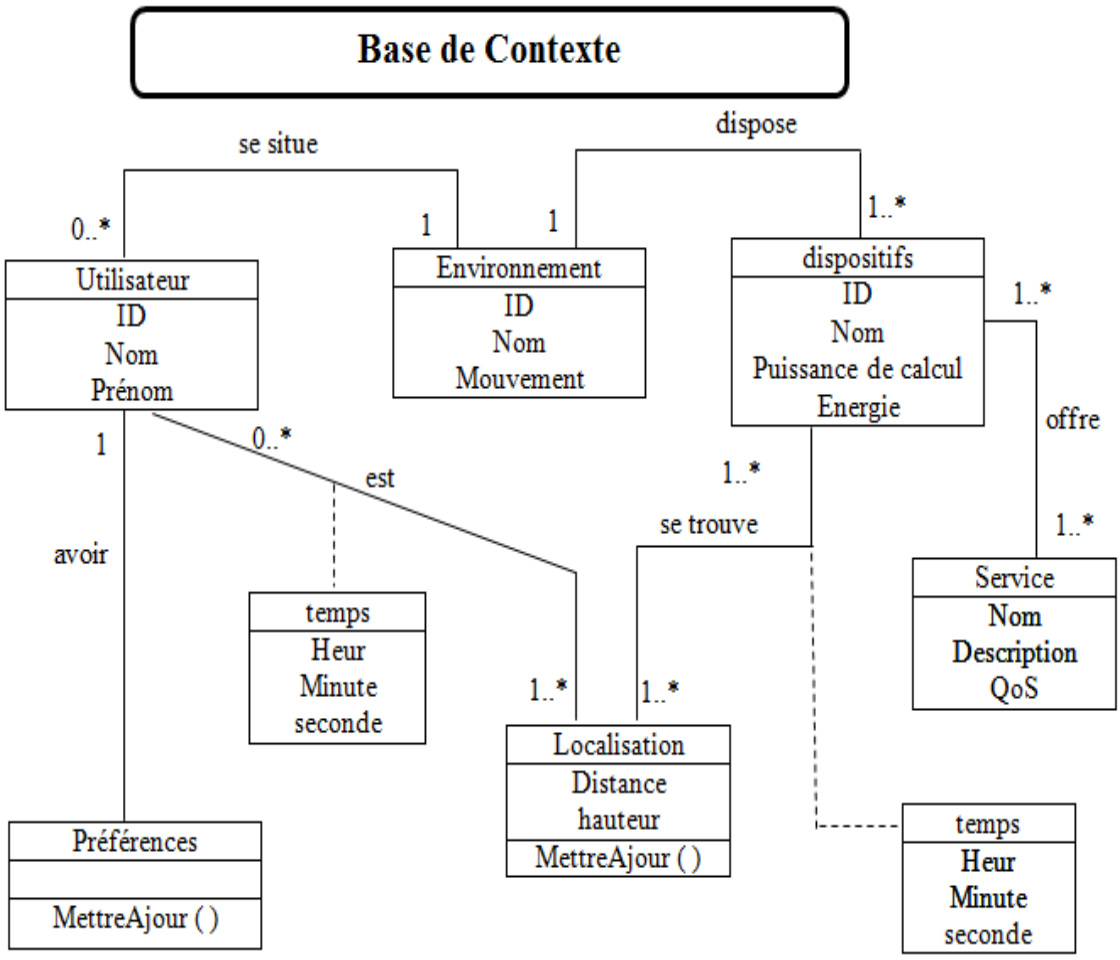


FIGURE 3.3 – La modélisation du contexte utilisée.

3.4.2 Attributs de contexte

Les attributs du contexte sont supposés être récupérés à partir des profils définis dans la modélisation du contexte (utilisateur, services, dispositifs ou environnement).

3.4.2.1 Énergie

L'énergie représente le niveau de batterie du dispositif qui hébèrge un service donné. Ce paramètre vise à vérifier si l'exécution du service durera jusqu'à sa fin [45]. Ce paramètre est mesuré en pourcentage. L'énergie est un attribut positif, il est donc à maximiser.

3.4.2.2 Localisation

La localisation permet de satisfaire l'utilisateur en s'adaptant à son contexte d'une manière transparente. L'objecctif de prendre en considération cet attribut est de sélectionner le dispositif (service) le plus proche à l'utilisateur pour répondre à sa requête (exemple, pour un service d'affichage sur l'écran, si l'utilisateur est à proximité de son PC la notification sera affiché sur l'écran de ce dernier).

Puisqu'on vise à minimiser la distance, on normalise alors la valeur de cet attribut (c.à.d, rendre sa valeur entre 0 et 1) par la formule :

$$d'_i = \begin{cases} \frac{d_{max}-d_i}{d_{max}-d_{min}} & \text{si} \quad d_{max} \neq d_{min} \\ 0 & \text{sinon} \end{cases} \tag{3.1}$$

Tel que d'_i est la valeur normalisée de la distance entre l'utilisateur et le dipositif.
d_{max} représente la distance maximale de la classe de services abstraits.
d_{min} représente la distance manimale da la classe de services abstraits.
d_i représente la distance entre le dipositif "i" (service) et l'utilisateur.

3.5 Qualité de service (QoS)

La qualité de service (notée QoS) désigne la capacité à fournir un service conforme aux exigences et aux préférences de l'utilisateur (en matière de temps de réponse par exemple). Dans cette section, nous détaillons la manière dont les paramètres de qualité de services sont calculés.

3.5.1 Temps de réponse

Le temps de réponse (noté TR) est le temps mis entre l'envoi de la requête par l'utilisateur et la réception du résultat envoyé par le service [43]. Le temps de réponse, appelé aussi latence, représente le temps requis par un service pour répondre à une demande (invocation) [7]. Dans notre travail, le temps de réponse est calculé en se basant sur les exécutions précédentes (voir l'algortihme 1).

Algorithm 1 Calcul du temps de réponse d'un service concret

Begin
$TR_total := 0$;
for $i := 1$ à n **do**
  $TR_i := T_{fin} - T_{debut}$;
  $TR_total = TR_total + TR_i$;
end for
$TR := TR_total/n$;
End

Tel que TR_{total} est le temps de réponse cumulé d'un service concret cs_i.
T_{debut} est le temps du début d'invocation du service concret cs_i.
T_{fin} est le temps de la fin d'invocation du service concret cs_i.
n : est le nombre d'invocations.

Pour chaque service concret, on calcule un délai d'attente $D_i = \alpha.TR_i \wedge \alpha > 1$, il représente le délai maximal à attendre pour recevoir la réponse du service cs_i. Au-delà de ce délai, la réponse est considérée inutile [43].

Algorithm 2 Délais max

Begin
$D_i := \alpha * TR$;
if délais d'attente $< D_i$ **then**
  Attente de réponse ;
else
  Arrêter d'attendre ;
end if
End

3.5.2 Fiabilité

La fiabilité (notée RL) représente l'aptitude d'un service à s'exécuter sans détériorer l'information traitée tout en respectant les demandes et les conditions contractuelles [2]. La fiabilité représente le taux du succès d'un service pour répondre à une invocation (voir l'algorithme 3).

Algorithm 3 Calcul de la fiabilité d'un service concret

Begin
$fiab_i := 0$;
for $i := 1$ à n **do**
  if la requête de l'utilisateur est satisfaite **then**
    $fiab_i := fiab_i + 1$;
  end if
end for
$RL := fiab_i/n$;
End

Tel que RL est la fiabilité de service sc_i.

3.5.3 Disponibilité

Ce paramètre prend une valeur entre 0 et 1. En effet, à un instant donné, un service peut être disponible (c.à.d, $dispo > 0$) ou indisponible (c.à.d, $dispo = 0$). L'algorithme 4 calcule la valeur de ce paramètre.

La disponibilité (notée AV) est un paramètre imprévisible car il dépend de plusieurs facteurs notamment le niveau d'énergie du dispositif de traitement, la mobilité du dispositif, la disponibilité des serveurs, l'état de la connexion réseau, etc.

Algorithm 4 Calcul de la disponibilité d'un service concret

Begin
$Nbdispo := 0$;
for $i := 1$ à n **do**
  if la requête de l'utilisateur est satisfaite **then**
    $Nbdispo := Nbdispo + 1$;
  end if
end for
$AV := Nbdispo/n$;
End

Tel que AV est la disponibilité du service cs_i, et $Nbdispo$ est le nombre de fois où le service est disponible.

3.5.4 Sécurité

Ce paramètre représente le degré de sécurité du service. La sécurité (notée SEC) vise à réaliser et à garantir les objectifs suivants : la confidentialité, l'authentification et l'intégrité. La sécurité est mesurée en fonction des objectifs qu'elle vise à atteindre, cela est estimé en utilisant l'algorithme 5.

Algorithm 5 Calcul de la sécurité d'un service concret

Begin
$vattr := 0$;
for $i := 1$ à 3 **do**
  if Le paramètre de sécurité est assuré **then**
    $vattr := vattr + 1$;
  end if
end for
$SEC := vattr/3$;
End

Tel que SEC est le degré de sécurité d'un service cs_i.
Dans le cas où $SEC = 1$, on pourra dire que le service concret sc_i est conforme aux trois objectifs de sécurité ciblés.

3.5.5 Prix

Cet attribut représente le coût monétaire que l'utilisateur doit payer pour utiliser une ressource, il est obtenu en compensant le coût du matériel et des logiciels. La plus part des fournisseurs fixent des prix différents pour leurs services.

Phase de mise à l'échelle Les attributs de QoS sont divisés en deux classes, les attributs positifs et les attributs négatifs. Les valeurs des attributs positifs doivent être maximisées (disponibilité, fiabilité et sécurité). En revanche, les valeurs des attributs négatifs doivent être minimisées (coût et temps de réponse).

Pour faire face à ce problème, une phase de mise à l'échelle doit être effectuée pour normaliser les valeurs des attributs de QoS.

La mise à l'échelle est une phase de prétraitement visant à normaliser les valeurs de QoS associées à des attributs positifs et négatifs, en les transformant en une valeur comprise entre 0 et 1.

La QoS d'un service candidat cs_i est représentée sous forme d'un vecteur $QoS(cs_i) =< q_{i,1}, q_{i,2}, ..., q_{i,n} >$, où n représente le nombre d'attributs de QoS requis par l'utilisateur et $q_{(i,j)}$ représente la valeur de l'attribut j $(1 \leq j \leq n)$ de QoS du service cs_i. Afin de normaliser les valeurs des attributs de QoS, on utilise les formules suivantes :

- ***Attributs négatifs :***

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

- ***Attributs positifs :***

$$q'_{(i,j)} = \begin{cases} \frac{q_{(i,j)} - q_j^{min}}{q_j^{max} - q_j^{min}} & \text{si} \quad q_j^{max} - q_j^{min} \neq 0 \\ 1 & \text{sinon} \end{cases} \tag{3.3}$$

$q'_{i,j}$ dénote la valeur normalisée de l'attribut j de QoS associé au service concret cs_i, elle est calculée en utilisant la valeur actuelle $q_{i,j}$ ainsi que les valeurs maximales et minimales de l'attribut j de QoS QoS_i qui se réfèrent respectivement aux q_j^{max} et q_j^{min}.

3.6 Apperçu sur les Algorithmes Génétiques

3.6.1 Présentation

Les algorithmes génétiques (notés AG) fournissent des solutions aux problèmes n'ayant pas de solutions calculables en temps raisonnable de façon analytique ou algorithmique. Les AG sont des approches d'optimisation qui utilisent des techniques dérivées de la science génétique et de l'évolution naturelle : la sélection, la mutation et le croisement et le remplacement.

Il est nécessaire de fournir une fonction permettant de coder une solution sous forme de gènes et une autre fonction pour évaluer la pertinence d'une solution au problème donné. Le principe d'un AG est simple, il s'agit de simuler l'évolution d'une population d'individus jusqu'à un critère d'arrêt.

Un AG doit disposer des cinq éléments suivants [46] :

- Principe de codage de l'élément de population : consiste à associer à chaque point de données de l'espace de l'élément, une structure de données.

- Une fonction de génération de la population initiale : ce mécanisme doit être capable de produire une population d'individus non homogènes, qui servira de base pour les générations futures.

- Une fonction à optimiser : une fonction d'utilité (fitness) pour classer les solutions en fonction de leurs aptitudes.

- Des opérateurs : définissent la manière dont les caractéristiques génétiques des parents sont transmises aux descendants (enfants). En effet, l'opérateur de croisement recompose les gènes d'individus existant dans la population et l'opérateur de mutation a pour but de garantir l'exploration de l'espace d'état.

- Les valeurs des paramètres utilisés par l'AG : la taille de la population, le nombre total de générations ou critères d'arrêt, la probabilité d'application des opérations de croisement et de mutation.

Un AG doit réspecter les critères de convergence suivants :

- Un taux minimum qu'on désire atteindre d'adaptation de la population au problème et un temps de calcul maximum à ne pas dépasser ;
- Une fonction d'utilité à maximiser.

3.6.2 Fonctionnement des algorithmes génétiques

Un algorithme génétique fontionne de la manière suivante [47] :

Étape 1 : Initialisation

On choisit N individus qui représentent la population initiale (P_0), ils sont tirés aléatoirement.

Étape 2 : Évaluation

On évalue chaque individu par la fonction d'utilité.

Étape 3 : Sélection

On définit les individus de la génération P_i, qui seront dupliqués dans la nouvelle population. À chaque génération, il y a deux opérateurs de sélection : la sélection de reproduction (ou plus simplement la sélection), déterminant les individus qui se reproduiront durant une génération et la sélection pour le remplacement (ou plus simplement le remplacement) qui détermine quels individus devront disparaître de la population.

Étape 4 : Reproduction

On utilise des opérateurs génétiques (croisement et mutation) pour produire la nouvelle génération. L'opérateur de mutation modifie un individu pour en former un autre tandis que l'opérateur de croisement engendre un ou plusieurs enfants à partir de combinaisons de deux parents.

Étape 5 : Retour

Retour à la phase d'évaluation tant que les conditions d'arrêt du problème ne sont pas atteintes.

3.6.3 Avantages et limites des algorithmes génétiques

Les AG présentent quelques avantages, parmi lesquels on cite [31] :

- Ils autorisent la prise en compte de plusieurs critères simultanément ;
- Ils parviennent à trouver de bonnes solutions sur des problèmes très complexes ;
- Ils combinent entre l'exploration de l'espace de recherche et l'exploitation des meilleures solutions disponibles à un moment donné ;
- Ils doivent simplement déterminer, entre deux solutions, quelle est la meilleure afin d'opérer leurs sélections.

Toutefois, les AGs ont des limites [31] :

- Le coût d'exécution important : les algorithmes génétiques nécessitent de nombreux calculs, en particulier au niveau de la fonction d'évaluation.

- L'ajustement d'un algorithme génétique est délicat : des paramètres tels que la taille de la population ou le taux de mutation sont parfois difficiles à déterminer. Or, le succès de l'évolution en dépend et plusieurs essais sont donc nécessaires, ce qui limite encore l'efficacité de l'algorithme.

- Lorsqu'une population évolue, il se peut que certains individus, qui à un instant occupent une place importante au sein de cette population, deviennent majoritaires.

- Le caractère indéterministe des algorithmes génétiques : comme les opérateurs génétiques utilisent des facteurs aléatoires, un algorithme génétique peut se comporter différemment pour des paramètres et populations identiques.

3.7 QSCGA : approche de composition de services basé sur les algorithmes génétiques

3.7.1 Hypothèses

Nous proposons une approche composition de services avec QoS basée sur les algorithmes géenéetiques (QSCGA) qui se base sur les hypothèses suivantes :

Hypothèse 1 : le modèle de découverte est chargé de trouver tous les services concrets disponibles qui sont enregistrés dans le répertoire des services concrets (RSC).

Hypothèse 2 : les services concrets existant dans RSC sont classés dans des services abstraits selon leurs fonctionnalités (entrées et sorties). Les services abstraits sont enregistrés dans un répertoire de service abstraits (RSA).

Hypothèse 3 : à chaque apparition ou disparition d'un service concret les deux répertoires (RSC et RSA) sont mis à jour.

Hypothèse 4 : le plan abstrait global est généré automatiquement en utilisant un algorithme de chainage arrière [2], et il est optimisé en se basant sur la réputation de chaque service abstrait pour éviter la redondance de services [37].

Quand un utilisateur éxprime ses besoins à travers une requête, elle sera traitée par le module de découverte qui éxplore les services concrets proposés par les fournisseurs et qui

sont enregistré dans RSC. Ensuite, ces services seront classés dans des services abstraits enregistrés dans RSA. Le contexte est traité par un intergiciel du contexte et stocké dans une base de contexte. Puis, une sélection (locale et globale) sera faite sur les services qui satisfont les exigences de l'utilisateur et s'adaptant à son contexte. Finalement, le plan concret sera exécuté selon l'ordre des services le constituant.

Dans la composition de services, généralement on distingue deux types de composition :

La composition abstraite : cette composition consiste à générer un plan abstrait global en se basant sur l'algorithme de chainage arrière [2], puis l'optimiser à base de la réputation des services abstraits [37].

La composition concrète : elle consiste à générer un plan de composition concret en se basant sur : (1) la sélection locale des services concrets en fonction de leurs QoS et du contexte (utilisateur et environnement) et (2) la sélection globale pour choisir le meilleur service composites parmi toutes les combinaisons possibles. Dans ce travail, on s'intéresse à ce type de composition tout en supposant l'existance d'un plan optimal abstrait.

3.7.2 Spécification des exigences de l'utilisateur

Les exigences de l'utilisateur doivent être bien explicites pour pouvoir les satisfaire. Ces exigences peuvent être locales (à satisfaire par chaque service concret) ou globale (à satisfaire par le service composite).

Au niveau local, l'utilisateur exprime ses exigences sous forme d'un seuil de qualité de service (QoS_{locale}) à satesfaire par chaque service concret, tel que tout service n'offrant pas cette qualité minimale sera rejeté du processus de composition.

Les exigences globales sont à satisfaire par le service composite. L'utilisateur spécifie donc ses exigences globales sur un ensemble d'attributs de QoS. Dans ce travail, on prend en compte tous les attributs de QoS (c. à. d, la qualité globale d'un service composite est une somme pondérée des valeurs de QoS des services le composant). En adoptant le principe des algorithmes génétiques, la qualité globale est définie par la fonction d'utilité.

Afin de sélectionner le meilleur service concret, on se base sur les attributs de QoS (temps de réponse, fiabilité, securité, disponibilité et coût)et le contexte (localisation et

niveau d'énergie). Cette sélection se déroule en deux étapes : (sélection locale et sélection globale).

3.7.3 Sélection locale

La sélection locale consiste à élire tous les services concrets dont la qualité d'expérience (QoE) dépasse le seuil (QoS_{locale}) spécifié par l'utilisateur [37]. La qualité de l'expérience QoE représente la qualité d'un service concret en fonction des préférences de l'utilisateur et du contexte. En d'autre terme, la QoE est une somme pondérée des paramètres de qualité de service (QoS) et du contexte. Le poids associé à chaque paramètre, est mesuré en fonction des préférences de l'utilisateur (spécifie dans son profil), son contexte et le contexte de l'environnement. Ce poids reflète le degré d'importance d'un attribut donné pour l'utilisateur.

Algorithm 6 Sélection locale de services

Begin
for tout AS du plan optimal **do**
  for tout $cs_i \in AS$ **do**
    $QoS_i = \sum_{j=1}^{NBQ}(w_j^k * q'_{(i,j)})$;
    $QoC_i = \sum_{l=1}^{NBC}(wc_l^k * qc_{(i,l)})$;
    $QoE_i = (QoS_i + QoC_i)/2$;
    if $QoE_i > QoS_{locale}$ **then**
      Service cs_i élu ;
    else
      Service cs_i rejeté ;
    end if
  end for
end for
End

Où NBQ représente le nombre de paramètres de QoS ;
NBC est le nombre de paramètres du contexte ;
QoS_i représente la qualité de service du service concret cs_i ;
QoC_i représente la qualité de contexte du service concret cs_i ;
QoE_i représente la qualité d'expérience du service concret cs_i ;

$q\prime_{i,j}$ représente la valeur normalisée de l'attribut "j" de QoS du service concret cs_i ;
$qc_{i,l}$ représente la valeur normalisée de l'attribut "l" du contexte ;
wc_l^k est le poid attribué par l'utilisateur "k" à l'attribut "l" du contexte ;
w_j^k représente le poids attribué par l'utilisateur "k" à l'attribut "j" de qualité de service ;

3.7.4 Sélection globale

Le plan obtenu de la phase précédente répond aux exigences de l'utilisateur localement. L'étape de sélection globale a comme objectif de vérifier la satisfaction des contraintes globales afin de sélectionner le meilleur service composite. En effet, lorsque les services concrets formant ce plan sont combinés ensemble, le service composite obtenu doit satisfaire la contrainte de la qualité globale (c.à.d, maximiser la fonction d'utilité).

L'approche proposée pour la sélection globale ce fait en adoptant le principe des algorithmes génétiques afin de sélectionner le meilleur service composite, elle est baptisée ***QSCGA*** (pour *QoS-aware Service Composition based Genetic algorithm*). Selon le modèle (schéma) de composition, nous associons à chaque paramètre de QoS considéré une valeur :

Attribut de QoS	Fonction d'agrégation			
	Séquentielle	Parallèle (ET)	Conditionnelle (XOR)	Boucle (LOOP)
Temps de réponse	$\sum_{j=1}^{n} TR_j$	$Max_{j=1}^{n} TR_j$	$Max_{j=1}^{n} TR_j$	$TR_j * k$
Disponibilité	$\prod_{j=1}^{n} Dispo_j$	$\prod_{j=1}^{n} Dispo_j$	$Min_{j=1}^{n} Dispo_j$	$(Dispo_j)^k$
Fiabilité	$\prod_{j=1}^{n} Fiab_j$	$\prod_{j=1}^{n} Fiab_j$	$Min_{j=1}^{n} Fiab_j$	$(fiab_j)^k$
Sécurité	$Min_{j=1}^{n} secur_j$	$Min_{j=1}^{n} secur_j$	$Min_{j=1}^{n} secur_j$	$secur_k$
Coût	$\sum_{j=1}^{n} cout_i$	$\sum_{j=1}^{n} cout_i$	$\sum_{j=1}^{n} cout_i$	$cout_i * k$

TABLE 3.1 – Fonction d'agrégation de QoS

- ***Séquentielle :*** une séquence de services $\{s_1, ..., s_n\}$ est exécutée dans un ordre chronologique strict.

- ***Boucle (LOOP) :*** un bloc d'un ou plusieurs services est exécuté de façon répétée jusqu'à k exécutions. Les valeurs agrégées de QoS d'une construction de boucle sont calculées en prenant la pire valeur parmi les k valeurs obtenues.

- ***Parallèle (ET) :*** plusieurs services $\{s_1, ..., s_n\}$ sont exécutés en même temps.

- ***Condition (XOR) :*** un ensemble de services $\{s_1, ..., s_n\}$ sont associés à une condition logique qui est évaluée à l'exécution et sur la base du résultat obtenu, un service est exécuté. Les valeurs de QoS estimées d'une construction conditionnelle sont les pires valeurs des services $\{s_1, ..., s_n\}$.

3.7.4.1 Principe d'application de l'algorithme génétique à la sélection globale

Dans l'approche proposée nommée QSCGA, les éléments de la population sont codés de la manière suivante :

- ***Le chromosome :*** représente un service composite qui est une chaine de gènes représentant les solutions potentielles.

- ***Gène :*** représente un service concret.

- ***La population :*** un ensemble de composition (caractérisé par des chromosomes) où les services composants sont sélectionnés aléatoirement.

- ***La fonction d'utilité :*** elle joue un rôle prépondérant dans le déroulement de l'algorithme génétique, elle sert à maximiser la valeur de qualité globale ($QoS_{globale}$) du service composite.

L'objectif ici est de trouver un service composite $Scomp = \{cs_1, cs_2, ..., cs_n\}$ tel que :

✓ Les valeurs des attributs de QoS de $Scomp$ sont maximisées ;
✓ Les contraintes globales de l'utilisateur sont satisfaites ;
✓ Le nombre maximale de générations est pas atteint ;

3.7.4.2 Description des étapes de QSCGA

1. Création de la population initiale

Dans cette étape, la population initiale est générée en sélectionnant aléatoirement N services composites parmi toutes les combinaisons possibles, ce nombre reste le même pour toutes les populations qui seront générées. Dans notre approche, nous proposons de calculer la valeur de N en devisant le nombre des combinaisons possibles (noté nbr_{comb}) sur le nombre de services abstraits (noté nbr_{AS}) se trouvant dans le plan optimal, suivant

la formule :

$$N = \frac{nbr_{comb}}{nbr_{AS}} \tag{3.4}$$

Le nombre de combinaisons est obtenu par la formule :

$$nbr_{comb} = \prod_{i=1}^{nbr_{AS}} |AS_i| \tag{3.5}$$

Où $|AS_i|$ représente le nombre de services concrets appartennant a la classe de service abstrait AS_i

2. Évaluation

Dans cette étape, on évalue chaque service composite $Scomp$ en calculant sa fonction d'utilité (notée F_i) et sa probabilité de sélection (notée $Prob_i$). Ensuite, on calcule la fonction d'évaluation de la population (notée EP_t). L'évaluation de la population est décrite par l'algorithme 7.

La fonction d'utilité F_i d'un service composite $Scomp_i$ est calculée en utilisant la formule suivante :

$$F_i = \sum_{j=1}^{NBQ} w_j * Q\prime_j \tag{3.6}$$

NBQ est le nombre de paramètre de QoS ;
w_j est le poids du j^{eme} attribut de QoS ;
$Q\prime_j$ est la valeur agrégée du j^{eme} attribut de QoS.

La probabilité de sélection $Prob_i$ d'un service composite $Scomp_i$ afin de lui appliquer par la suite l'une des opérations génétiques, est obtenue par la formule :

$$Prob_i = \frac{F_i}{\sum_{j=1}^{N} F_j} \tag{3.7}$$

La fonction d'évaluation EP_t de la population actuelle représentant l'évolution de cette population, est obtenue par la formule :

$$EP_t = \frac{\sum_{i=1}^{N} F_i}{N} \tag{3.8}$$

Algorithm 7 Évaluation de la population t

Begin
for tout $Scomp_i$ de la population **do**
$F_i = \sum_{j=1}^{NBQ} w_j * Q\prime_j$;
$Prob_i = \frac{F_i}{\sum_{j=1}^{N} F_j}$;
end for
$EP_t = \frac{\sum_{i=1}^{N} F_i}{N}$;
End

3. Sélection par rang

Elle conciste à sélectionner le meilleur service composite en termes de la probabilité de sélection pour une opération génétique. On a deux cas :

- Le cas où l'opération génétique est une mutation, la sélection retourne alors un seul service (parent) qui a la $Prob_i$ maximale.

- Le cas où l'opération génétique est un croisement, la sélection retourne alors deux services composite ($parent_1$ et $parent_2$) ayant la $Prob_1$ et $Prob_2$ respectivement representant les deux meilleures probabilités de la population. Le service composite qui a la probabilité maximale sera sélectionné comme $parent_1$ et l'autre sera sélectionné comme $parent_2$ (voir algorithme 8).

Algorithm 8 Sélection par rang

Begin
if opération est une mutation **then**
Sélectionner $parent = ArgMax(Prob_i)$;
else
//opération est un croisement
Sélectionner $parent_1 = ArgMax(Prob_i)$ dans la popualtion P ;
$P' = P - parent_1$;
Sélectionner $parent_2 = ArgMax(Prob_i)$ dans la popualtion P' ;
end if
End

La fonction $ArgMax(Prob_i)$, retourne le service composite qui a la probabilité maximum dans la population.

4. Application des opérations génétiques

✓ **Mutation :** cette opération est illustrée par l'algorithme 9. Elle consiste à changer, dans le service parent (service composite), le service concret possédant la QoE minamale (noté sc_{min}) par un autre service (noté sc_{max}) ayant une meilleure QoE et appartenant au même service abstrait. La fonction $ArgMin(QoE_i)$ retourne le service sc_{min} tandis que la fonction $ArgMax(QoE_i)$ retourne le service sc_{max}.

Algorithm 9 Opération de mutation

Begin
Pour le service parent ;
$sc_{min} = ArgMin(QoE_i)$;
Pour le service abstrait auquel sc_{min} appartient
$sc_{max} = ArgMax(QoE_i)$
Remplacer sc_{min} par sc_{max} dans le service parent ;
End

✓ **Croisement :** le croisement consiste en une opération entre deux service composites. Dans notre approche de composition, le croisement se fait entre les deux meilleurs services composites de la manière suivant l'algorithme 10.

Algorithm 10 Opération de croisement

Begin
for $i = 1$ à $|parent_1|$ **do**
  if $QoE_i(parent_1) \leq QoE_i(parent_2)$ **then**
    permuter$(sc_i(parent_1, sc_i(parent_2))$;
  end if
end for
End

Il est a noté que $QoE_i(parent_1)$ représente la QoE du service concret sc_i du $parent_1$ et la fonction $permuter(sc_i(parent_1), sc_i(parent_2))$ permute entre deux services concrets qui appartiennent à deux services composites différents ($parent_1$ et $parent_2$).

✓ **Remplacement :** l'objectif de cette opération est de substituer des services de la population par ceux obtenus en applicant les opérations de mutation et de croisement. Dans cette étape, on a deux cas :

- L'opération est un croisement, si la valeur de F de l'un des fils est supérieur à celle de l'un des parents (tel que $F(parent_1) \geq F(parent_2)$), alors on remplace $parent_2$ par le fils en question.

- L'opération est une mutation, si la valeur de F du fils est supérieur à celle du parent, on remplace le service composite qui a la valeur minimale de F par le fils. On fait le même remplacement dans le cas contraire.

Algorithm 11 Remplacement

Begin
//Remplacement si opération est un croisement
if $F(fils) > F(parent_1)$ **then**
  $Remplacer(parent_2, fils)$;
  $nbr_{cR}++$;
else
  if $F(fils) > F(Parent_2)$ **then**
    $Remplacer(parent_2, fils)$;
    $nbr_{cN}++$;
  else
    $nbr_{cN}++$;
  end if
end if
//Remplacement si opération est une mutation
if $F(fils) > F(parent)$ **then**
  $S_r = ArgMin(F_i)$;
  Remplacer$(S_r, fils)$;
  $nbr_{mR}++$;
else
  Remplacer$(S_r, fils)$;
  $nbr_{mN}++$;
end if
End

Où nbr_{cR} et nbr_{mR} sont le nombre de fois qu'une opération de croisement ou de mutation a réussi, c.à.d. la valeur de F du fils est supérieure à celle du parent ;

nbr_{cN} et nbr_{mN} sont le nombre de fois qu'une opération de croisement ou de mutation a échoué, c.à.d. la valeur de F du fils est inférieure à celle du parent ;

S_r est le service composite obtenu de la fonction $ArgMin(F_i)$ qui renvoie le service ayant la valeur minimal de F.

Après l'application des opérations génétiques sur la population (P_t), on obtient une nouvelle population ($P_{(t+1)}$), puis aller à l'étape 2 (évaluation).

3.7.4.3 Nombre maximum de générations

Le nombre maximal de générations, noté $nbr_{gen_{max}}$, définit le nombre maximal de fois où on a effectué les opérations génétiques (mutation ou croisement). Dans [48, 49], le nombre de descendants pour chaque chromosome (service composite) est éstimé par la formule suivante :

$$nbr_{descendant_i} = prob_i * N \tag{3.9}$$

On s'est inespéré de cette formule pour calculer le nombre maximal de générations. En effet, il est obtenu en multipliant le nombre de services composites de la population par la probabilité maximale dans cette population :

$$nbr_{gen_{max}} = N * max_{i=1}^{N}(prob_i) \tag{3.10}$$

3.7.4.4 Critères d'arrêt

L'approche proposée basée sur les algorithmes génétiques s'arrête dans l'un des cas suivants :

- Atteindre le nombre maximal de générations ($nbr_{gen_{max}}$) ;

- Atteindre un état de stabilité, c.-à-d. la valeur de $EP_{(t+1)}$ (population actuelle) est proche de celle de EP_t (population précédente) :

$$|EP_{(t+1)} - EP_t| \leq \varepsilon. \tag{3.11}$$

  Où EP_t représente la valeur de la fonction d'évaluation de la population t, $EP_{(t+1)}$ est la valeur de la fonction d'évaluation de la population $t+1$ et ε repésente une constante générée aléatoirement, tel que $0 \leq \varepsilon < 1$.

3.7.4.5 Organigramme de sélection globale de l'approche QSCGA

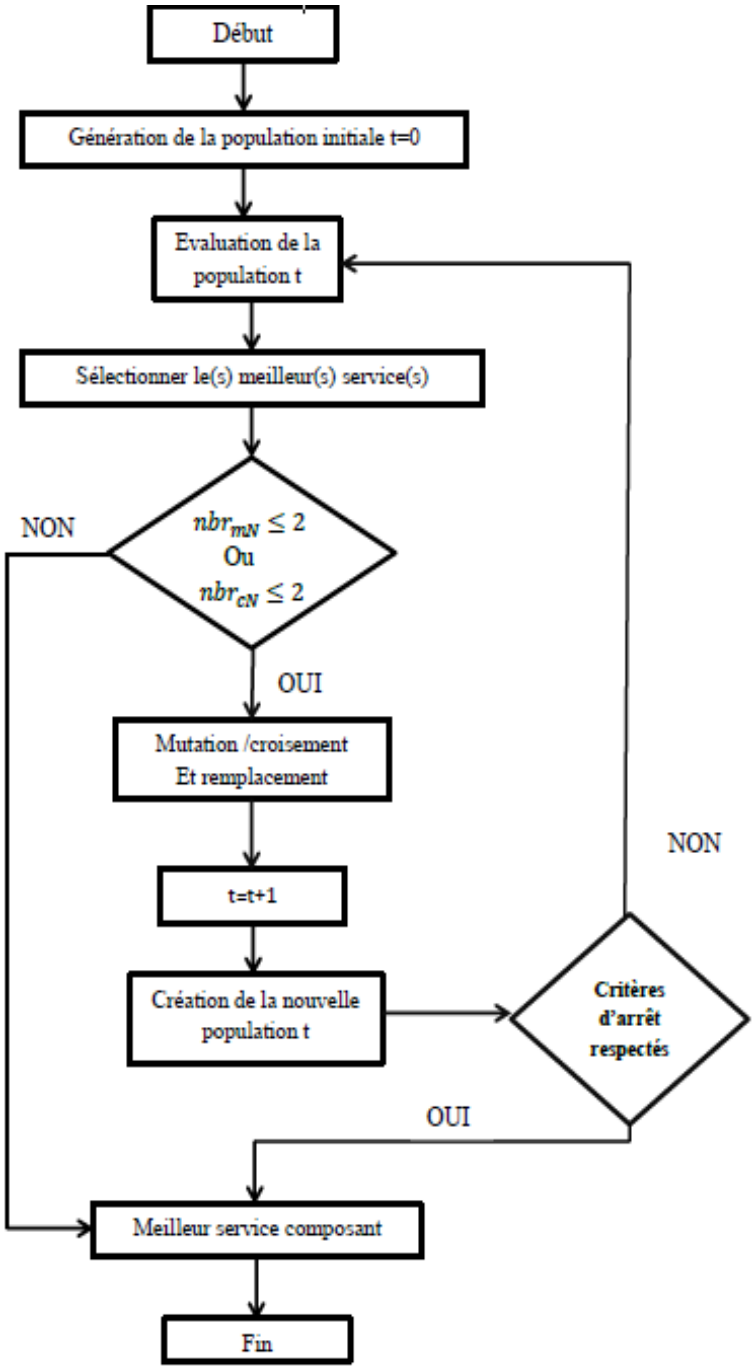


FIGURE 3.4 – Organigramme de sélection globale de services.

3.7.5 Contrôle et exécution des services

Après la sélection des services concrets les plus appropriés vérifiant les conditions exigées dans les deux phases de notre approche (sélection locale et globale), le plan d'exécution sera exécuté par la suite selon un ordre établi. Le contrôle du plan d'exécution est introduit à ce niveau, il permet de garantir une adaptation automatique et une tolérance aux défaillances pouvant survenir durant l'exécution, et ce en remplaçant le service concret qui a échoué par un autre service réalisant la même fonctionnalité. Dans certaines situations, le remplacement devient inopérant en cas d'absence du service concret supposé pallier le problème de défaillance, alors le plan optimal initial sera localement mis à jour à partir du plan global abstrait, cela permet d'éviter la phase de redécouverte de services et la régénération d'un nouveau plan global.

Dans le cas où l'exécution du plan de composition a réussi, le service composite sera publié dans le répertoire (CSD : Concret Service Directory) pour une utilisation ultérieure, et les valeurs de QoS sont mise à jour.

3.8 Scénario d'application

Afin d'illustrer le déroulement de l'approche proposée nous utilisons le scénario représentant un hôpital intelligent inspéré de [4]. Il s'agit particulièrement de surveiller le mouvement des patients dans cet hôpital et de transmettre ensuite les informations nécessaires aux fournisseurs de service.

L'hôpital est choisi comme un environnement pervasif disposant d'un certain nombre de dispositifs pour offrir des services aux patients, parmi lesquels on cite :

- Bracelet-montre : un bracelet-montre émet des alertes radiofréquences dues à la détection des alertes de types :
  - Alerte manuelle par appui sur un bouton d'urgence (ex. le patient appuie sur le bouton d'urgence s'il sent un malaise).
  - Détection d'un pouls hors norme : à partir de la mesure périodique du pouls et de la valeur moyenne fixée à l'initialisation, le bracelet peut émettre une alerte.
  - Détection de chute : le bracelet permet de détecter une chute à partir d'un profil

d'activité fixé à l'initialisation.

- Caméras de surveillance : elle se charge de prendre des photos de patients en cas de nécessité, (ex. en cas de chute du patient, la caméra prend une photo pour montrer la position du patient à son médecin).

- Des dispositifs d'affichage : téléphone portable de type smartphone, moniteur de PC, etc.

3.8.1 Capteurs utilisés dans le scénario

Le capteur est un dispositif transformant l'état d'une grandeur physique et ou logique observée en une grandeur utilisable. Dans le tableau ci-dessous on cite les capteur utilisés dans l'environnement :

Capteur	**Description**
Détecteur de présence	Ce capteur sert à détecter si un espace est occupé ou non (ex. il est utilisé dans les chambres d'hôpital pour déduire si le patient est dans sa chambre ou non).
Détecteur de chute	Ce capteur permet de déduire qu'une personne a tombé.
Identificateur	Ce capteur permet d'identifier une entité dans l'environnement (personne ou objet).
Système de localisation	Ce capteur permet de localiser une personne.
Capteur d'humidité	Ce capteur sert à capturer le taux d'humidité dans un endroit donné, ces informations permettent de savoir si la personne pourra se trouver dans cet endroit ou non.
Capteur de température	Ce capteur est nécessaire pour déterminer la température dans un endroit donné, pour permettre l'activation de la climatisation ou le chauffage.
Détecteur d'activation d'alarme d'urgence	Ce capteur est pour déduire si la personne est en situation d'urgence.
Détecteur de luminosité	Ce capteur permet de mesurer le taux de luminosité dans une chambre (ex. dans la journée si le patient est dans sa chambre la luminosité doit être moyenne).

TABLE 3.2 – Scénario applicatif : description des dispositifs existants dans l'hôpital

3.8.2 Description du scénario

Notre scénario consiste en l'assistance des patients dans un hôpital intelligent. John est le patient qui a l'identifiant "015", ce patient porte un bracelet-montre, qui a le rôle

de détecteur de chute, identificateur, et système de localisation. John décide de sortir au jardin de l'hôpital pour profiter du soleil. Soudainement, John tombe par terre, le détecteur de chute détecte que le patient est par terre. Suite à cet évènement, on aura deux sorties :

- Le détecteur de chute déclenche la caméra de surveillance pour envoyer un flux vidéo au serveur centrale de l'hôpital ;

- Le détecteur de chute déclenche le service de notification pour envoyer une notification à ce serveur, qui à son tour récupère l'identifiant de John, afin de collecter les informations nécessaires sur le patient (nom du médecin traitant et l'infirmière responsable) et leurs envoyer une notification.

Le docteur Stev est dans une réunion pour cela la notification s'affichera directement sur l'écran de sonordinateur, et l'infirmière Carla reçoit la notification sur son PDA. Stev interroge le système pour lui envoyer les informations nécessaire sur John (rythme cardiaque, son état : conscient / inconscient, et sa position). Le système après avoir reçu la requête de Stev, il la traite pour pouvoir lui répondre. Stev à son tour quand il reçoit la réponse de sa requête, il envoie un message pour informer l'infirmière de ce qu'elle doit faire si ça nécessite que les premiers soins, ou qu'est-ce qu'elle doit lui préparer le temps de son arrivée. Les services abstraits sont :

AS	**Entrées**	**Sorties**	**Description**
AS_1	$e_{1,1}$	$s_{1,1}$	Analyse du mouvement de l'environnement
AS_2	$s_{5,1}, e_{2,1}$	$s_{2,1}$	localisation du patient
AS_3	$e_{3,1}, s_{5,1}$	$s_{3,1}, s_{3,2}$	localisation du médecin
AS_4	$e_{4,1}$	$s_{4,1}$	Détecteur de mouvement du patient
AS_5	$s_{9,1}$	$s_{5,1}$	Service de notification
AS_6	$s_{9,1}, s_{6,1}$	$s_{6,1}$	Historique des soins
AS_7	$e_{7,1}$	$s_{7,1}$	Analyse de l'activité cardiaque
AS_8	$s_{2,1}$	$s_{8,1}$	Caméra de surveillance
AS_9	$s_{7,1}, s_{4,1}, s_{11,1}$	$s_{9,1}$	Détection de l'état du patient
AS_{10}	$e_{10,1}$	$s_{10,1}, s_{10,2}, s_{10,3}$	Capteur de météo
AS_{11}	$e_{11,1}$	$s_{11,1}$	Détection d'activation d'alarme d'urgence

TABLE 3.3 – Scénario applicatif : les services abstraits

Dans le tableau 4 on donne description des entrées / sorties des services abstraits :

Entrées	**Description**	**Sorties**	**Description**
$e_{1,1}$	Vecteur de données (température, humidité, luminosité)	$s_{1,1}$	Vecteur de données analysées
$e_{2,1}$	Signale de localisation	$s_{2,1}$	Localisation du patient
$e_{3,1}$	Signale de localisation	$s_{3,1}$	Localisation du médecin
$s_{3,2}$		le temps nécessaire pour l'arrivée du médecin a la localisation du patient	
$e_{4,1}$	Signal de chute du patient	$s_{4,1}$	Détection de la chute
$s_{5,1}$		notification qui s'affichera sur : TV, PC, PDA, tablette, etc.	
$e_{6,1}$	Historique des soins de la personne (vecteur des soins)	$s_{6,1}$	Vecteur des soins analysés (urgence, ou pas)
$e_{7,1}$	Signale cardiaque	$s_{7,1}$	Analyse du signale
$s_{8,1}$		Flux vidéo	
$s_{9,1}$		Vecteur de données sur l'état du patient	
$e_{10,1}$	Vecteur de données {température, vent}	$s_{10,1}$	Vecteur de données {Ensoleillé, Pluie, Vent}

TABLE 3.4 – Scénario applicatif : description des entrées sorties des services abstrait

Dans notre scénario la requête de l'utilisateur a comme entrées rythme cardiaque, état du patient, position, état climatique, historique des soins, et comme sortie notification, flux vidéo, $R^{in} = \{e_{2,1}, e_{7,1}, e_{6,1}, e_{11,1}, e_{4,1}\}$ et $R^{out} = \{s_{6,1}, s_{5,1}, s_{8,1}\}$.

3.8.3 Application des étapes de l'approche proposée au scénario

3.8.3.1 Génération du plan optimal

Le plan abstrait de notre scénario est obtenu en appliquant un algorithme de chinage arrière qui a comme entrées R^{in}, et sorties R^{out}, en optimisant le plan obtenu de cet algorithme par garder que les services abstraits qui ont une réputation élevée en respectant la requête de l'utilisateur.

Le plan optimal est le suivant :

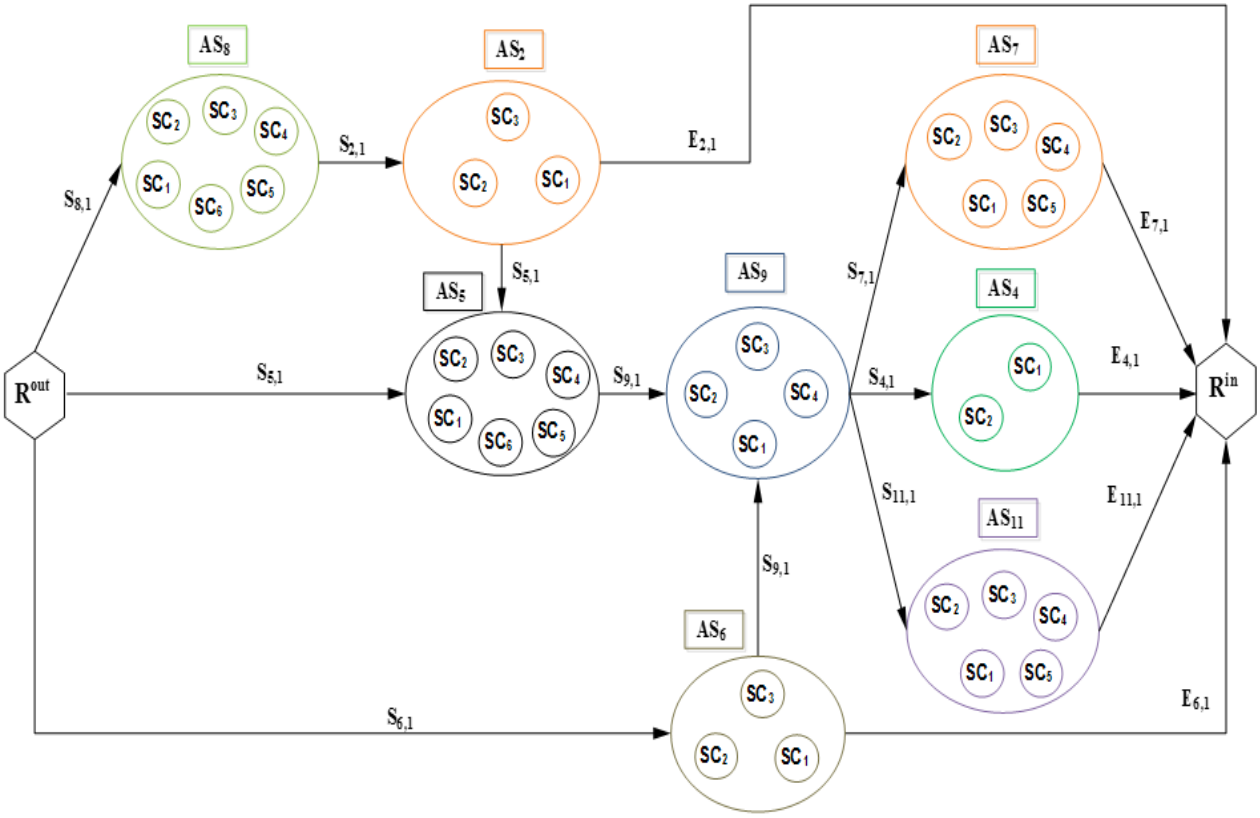


FIGURE 3.5 – Scénario applicatif : plan abstrait optimal de la composition de services

3.8.3.2 Sélection locale des services concrets

La sélection locale des services se fait à base de la QoE, on ne garde que les services qui ont la $QoE \geq QoS_{locale}$, les préférences de l'utilisateur en terme de QoS sont 0.5 pour le temps de réponse (TR), 0.3 pour la fiabilité (fiab) et 0.2 pour la sécurité (sec). Et en terme de contexte sont 0.4 pour la localisation (d) et 0.6 pour l'énergie (energ). Le seuil local spécifié est 0.6.

AS_i	Service concret	TR	sec	fiab	dispo	coût	energ	d	QoS	QoC	QoE
AS_2	$cs_{2,1}$	0,9	0	0,6	0,69	0,54	0,55	0,65	0,63	0,59	0,61
	$cs_{2,2}$	0,6	0,33	0,35	0,14	0,2	0,69	0,36	0,47	0,56	0,51
	$cs_{2,3}$	0,72	1	0,62	0,8	0,6	0,7	0,57	0,75	0,65	0,70
AS_4	$cs_{4,1}$	0,8	0,66	0,85	0,24	0,35	0,68	0,85	0,79	0,75	0,77
	$cs_{4,2}$	0,63	0,33	0,45	0,58	0,52	0,71	0,39	0,52	0,58	0,55
AS_5	$cs_{5,1}$	0,54	0,66	0,41	0,3	0,26	0,3	0,26	0,53	0.28	0,40
	$cs_{5,2}$	0,68	0,33	0,19	0,51	0,63	0,48	0,42	0,46	0,46	0,46
	$cs_{5,3}$	0,86	0,33	0,59	0,25	0,45	0,9	0,59	0,67	0,78	0,72
	$cs_{5,4}$	0,75	0,66	0,47	0,35	0,4	0,6	0,85	0,648	0,7	0,67
	$cs_{5,5}$	0,4	0	0,63	0,4	0,8	0,74	0,27	0,39	0,55	0,47
	$cs_{5,6}$	0,6	0	0,29	0,5	0,6	0,5	0,64	0,39	0,56	0,47
AS_6	$cs_{6,1}$	0,7	1	0,23	0,6	0,27	0,67	0,52	0,62	0,61	0,62
	$cs_{6,2}$	0,2	0,33	0,89	0,8	0,84	0,4	0,68	0,43	0,51	0,47
	$cs_{6,3}$	0,88	0,33	0,78	0,9	0,59	0,86	0,45	0,74	0,7	0,72
AS_7	$cs_{7,1}$	0,26	0,66	0,65	0,4	0,68	0,3	0,5	0,46	0,38	0,42
	$cs_{7,2}$	0,37	1	0,27	0,1	0,85	0,27	0,41	0,47	0,33	0,39
	$cs_{7,3}$	0,5	1	0,8	0,8	0,3	0,74	0,8	0,69	0,76	0,73
	$cs_{7,4}$	0,8	0	0,5	0,65	0,4	0,76	0,3	0,55	0,58	0,56
	$cs_{7,5}$	0,69	0,66	0,45	0,24	0,6	0,49	0,9	0,61	0,65	0,63
AS_8	$cs_{8,1}$	0,54	0,33	0,39	0,5	0,65	0,43	0,58	0,45	0,49	0,47
	$cs_{8,2}$	0,27	0,66	0,7	0,89	0,47	0,24	0,2	0,48	0,22	0,35
	$cs_{8,3}$	0,56	0,66	0,61	0,26	0,58	0,6	0,41	0,59	0,52	0,56
	$cs_{8,4}$	0,85	0	0,8	0,47	0,3	0,63	0,72	0,67	0,67	0,67
	$cs_{8,5}$	0,23	0	0,26	0,39	0,75	0,6	0,69	0,62	0,64	0,63
	$cs_{8,6}$	0,7	0,33	0,67	0,2	0,14	0,47	0,52	0,19	0,49	0,34
AS_9	$cs_{9,1}$	0,7	0,66	0,35	0,4	0,47	0,8	0,32	0,59	0,61	0,59
	$cs_{9,2}$	0,3	0	0,25	0,28	0,28	0,27	0,4	0,23	0,32	0,27
	$cs_{9,3}$	0,45	1	0,54	0,69	0,8	0,36	0,5	0,59	0,42	0,50
	$cs_{9,4}$	0,49	1	0,75	0,45	0,6	0,5	0,63	0,67	0,55	0,61
AS_{11}	$cs_{11,1}$	0,2	0	0,83	0,23	0,43	0,3	0,3	0,35	0,3	0,32
	$cs_{11,2}$	0,82	0,33	0,5	0,68	0,72	0,8	0,77	0,63	0,79	0,71
	$cs_{11,3}$	0,54	0,66	0,24	0,8	0,6	0,75	0,32	0,47	0,58	0,53
	$cs_{11,4}$	0,49	0,66	0,8	0,7	0,8	0,52	0,85	0,62	0,65	0,63
	$cs_{11,5}$	0,24	0,33	0,7	0,36	0,4	0,49	0,25	0,4	0,39	0,4

TABLE 3.5 – Scénario applicatif : les valeurs des attributs de QoS pour chaque service concret

La figure 6. montre le plan obtenu en appliquant l'algorithme de sélection locale :

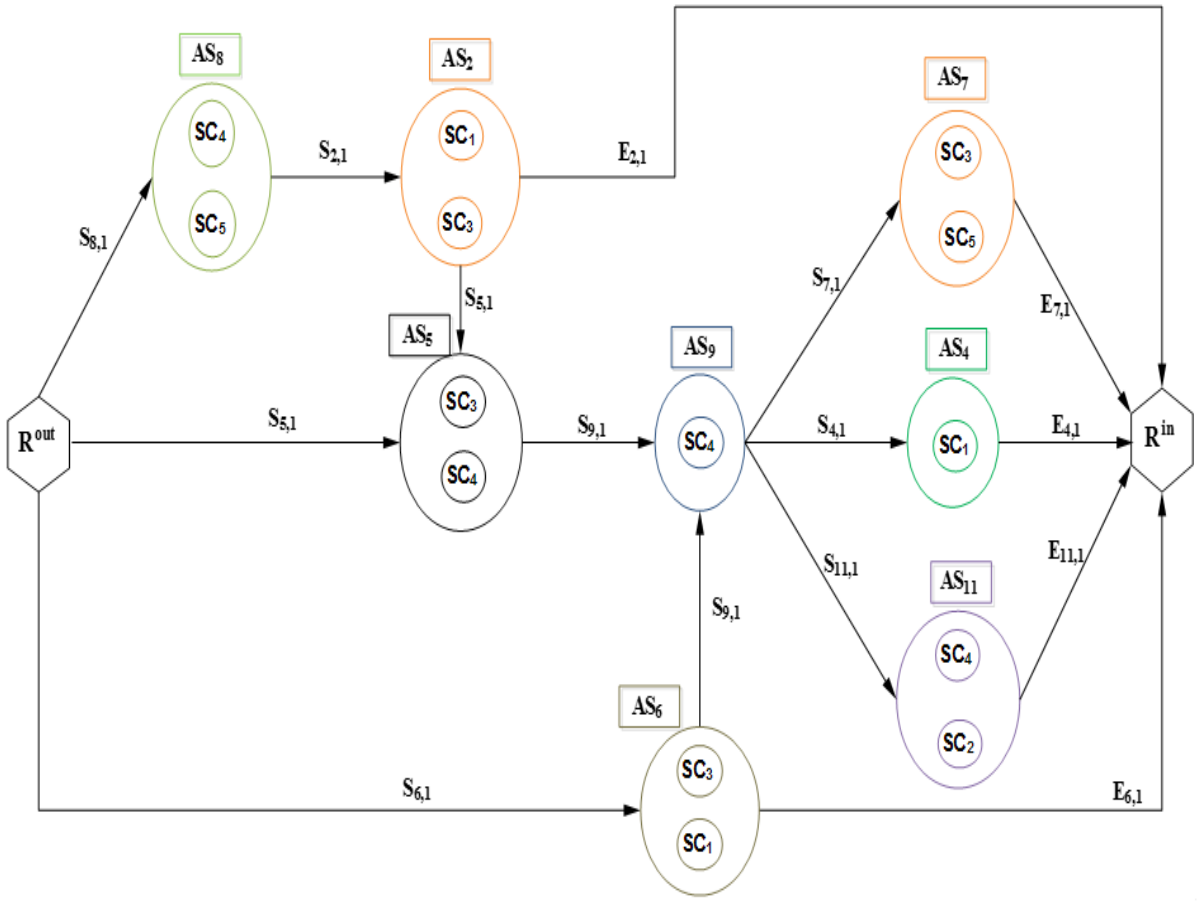


FIGURE 3.6 – Scénario applicatif : plan obtenu de la sélection locale

3.8.3.3 Sélection globale par l'approche QSCGA

Initialisation

$nbr_comb = |AS_2| * |AS_4| * |AS_5| * |AS_6| * |AS_7| * |AS_8| * |AS_9| * |AS_{11}| = 64.$

Le nombre de combinaison dans la population initiale N=8.

On sélectionne aléatoirement huit composition parmi celles possibles, la figure 7 montre les services composites sélectionnés pour former la population P_0.

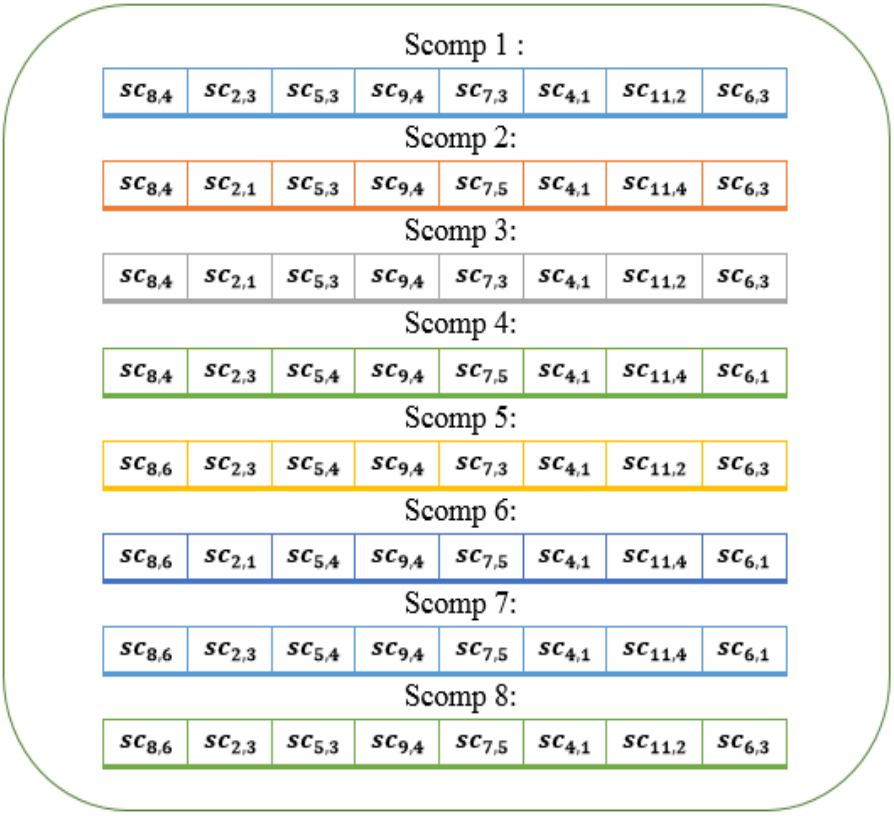


FIGURE 3.7 – Scénario applicatif : Les services composites sélectionés pour former la population P_0

Calcul des valeurs agrégées des services composites

L'ordre d'exécution est le suivant : $(AS_8, AS_2, (AS_5||AS_6), AS_9, (AS_7||AS_4||AS_{11}))$. Le tableau représente les valeurs d'attributs de QoS, les valeurs de la fonction d'utilité (F), la probabilité (prob) et l'évaluation de la population (EP).

$Scomp-i$	attributs de QoS					F_i	$prob_i$	**EP**(P_0)
	TR_i	$securite_i$	$fiab_i$	$dispo_i$	$cout_i$			
$scomp_1$	3,42	0	0,06	0,005	3,91	1,73	0,125	1,73
$scomp_2$	3,59	0	0,05	0,001	4,21	1,81	0,131	
$scomp_3$	3,6	0	0,06	0,004	3,83	1,82	0,132	
$scomp_4$	3,25	0	0,01	0,001	3,92	1,63	0,118	
$scomp_5$	3,16	0,33	0,04	0,002	3,7	1,66	0,12	
$scomp_6$	3,28	0	0,01	0	3,68	1,64	0,119	
$scomp_7$	3,3	0,33	0,01	0	3,76	1,72	0,125	
$scomp_8$	3,46	0,33	0,03	0	4,05	1,80	0,131	
						13,81		

TABLE 3.6 – Scénario applicatif : les valeurs de QoS des services composites de la population P_0

Le nombre de génération maximum éstimé par la formule suivante : $nbr_{gen_{max}} = max(prob_i) * N$ $nbr_{gen_{max}} = 1$

Première itération

Le service composite $Scomp_3$ est sélectionné car $max(prob_i) = prob_3$, on applique l'opération de mutation comme le montre la figure 8.

FIGURE 3.8 – Scénario applicatif : L'opération de mutation éffectuée sur $Scomp_3$

Le service composite obtenu en mutant $Scomp_3$ est $fils_3$, qui est identique à $scomp_1$. Pour cela on doit choisir un autre service composite à muter. Le nombre de mutation non réussie est incrémenté ($nbr_{mN} = 1$).

Le $Scomp_2$ est sélectionné car il possède une probabilité maximum dans la population P_0 et $Scomp_2 \neq Scomp3$, on applique l'opération de mutation sur $Scomp_2$.

Deuxième itération

Dans cette itération on ne peut pas choisir le service composite $Scomp_3$ car il n'existe pas un autre service concret apart $sc_{2,1}$ à muter appart $sc_{2,1}$, ce qu'est déjà fait. Pour

cela $Scomp_2$ sera sélectionné car il possdède une probabilité de sélection maximale après $Scomp_3$.

FIGURE 3.9 – Scénario applicatif : l'opération de mutation éffectuée sur $Scomp_2$

Le service composite obtenu en mutant $Scomp_2$ est $fils_2$. Après l'évaluation de $fils_2$ on trouve que $F(Scomp_2) > F(fils_2)$, on sélectionne le service qui a la probabilité minimale dans la population P_0, qui est $Scomp_4$ et on le remplace par $fils_2$.

Les résultats obtenus de l'évaluation de la population P_1 sont montrés dans le tableau suivant :

$Scomp_i$	attributs de QoS					F_i	$prob_i$	$\mathbf{EP}(P_1)$
	TR_i	$securite_i$	$fiab_i$	$dispo_i$	$cout_i$			
$scomp_1$	3,42	0	0,06	0,005	3,91	1,73	0,12	1,74
$scomp_2$	3,59	0	0,05	0,001	4,21	1,81	0,13	
$scomp_3$	3,6	0	0,06	0,004	3,83	1,82	0,13	
$fils_2$	3,41	0	0,05	0,002	4,29	1,72	0,12	
$scomp_5$	3,16	0,33	0,04	0,003	3,7	1,66	0,12	
$scomp_6$	3,28	0	0,01	0,001	3,68	1,64	0,12	
$scomp_7$	3,3	0,33	0,01	0,001	3,76	1,72	0,12	
$scomp_8$	3,46	0,33	0,03	0,001	4,05	1,80	0,13	
						13,90		

TABLE 3.7 – Scénario applicatif : les valeurs de QoS des services composites de la population P_1

La figure 10 montre les services composites qui sont sélectionnés pour former la population P_1 :

FIGURE 3.10 – Scénario applicatif : Les services composites sélectionnés pour former la population P_1

Le remplacement du service composite $Scomp_4$ de la population P_0 par le service composite obtenu de la mutation fil_2, donne naissance à une nouvelle génération ($nbr_{gen} = 1$), mais la valeur de fonction d'utilité de ce dernier est inférieur à celle de son parent, dans ce fait on incrémente le nombre de mutation non réussie ($nbr_{mN} = 2$).

On remarque que $nbr_{gen} < nbr_{gen_{max}}$ et $|EP(P_1) - EP(P_0)| \geq \varepsilon = 0.01$).

Dans ce scénario "ε" est choisi très petit car on a un nombre de génération maximum ($nbr_{gen_{max}}$ petit.

Troisième itération

Dans l'itération précédente on a obtenu $nbr_{mN} = 2$ pour cela dans cet itération on éffectue l'opération de croisement.

Les deux meilleur services composites dans la population P_1, qui sont sélectionnés pour le croisement, sont $Scomp_2$ et $Scomp_3$.

En appliquant l'algorithme de croisement sur les deux services composites sélectionés, on obtiendra pas de résultat car il n'existe pas de services concrets ayant une QoE dans $Scomp_3$ inférieur à la QoE d'un service concret dans $Scomp_2$.

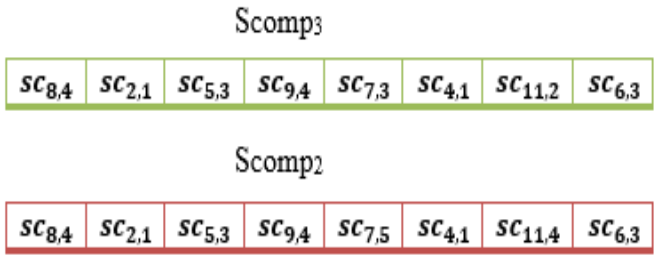


FIGURE 3.11 – Scénario applicatif : Les services composites sélectionnés pour le croisement

Le nombre de croisements non réussis s'incrémente : $nbr_{cN} = 1$, donc on choisit d'autre services composites pour le croisement.

Quatrième itération

On effectue le croisement entre le meilleur service composite ($Scomp_3$) et le troisième en terme de probabilité de sélection ($Scomp_8$).

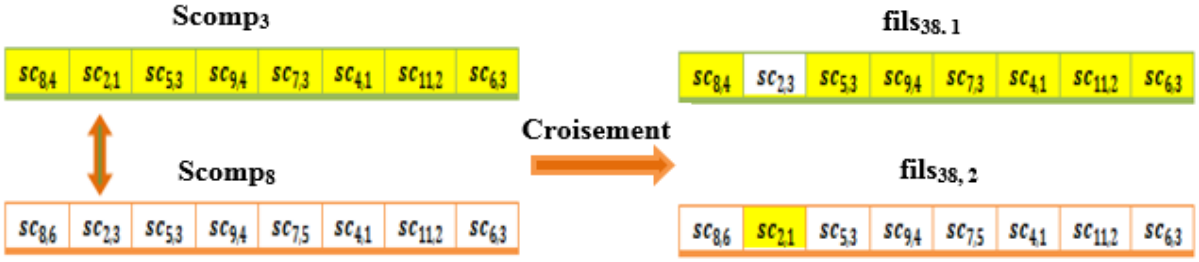


FIGURE 3.12 – Scénario applicatif : l'opération du croisement éffectuée sur $Scomp_3$ et $Scomp_8$

Le $fils_{38,1}$ est identique à $Scomp_1$, donc on calcule que la fonction d'utilité (F) du service composite $fils_{38,2}$. On trouve que $F(fils_{38,2}) \geq F(Scomp_3)$. On remplace le parent ($Scomp_8$) qui a la valeur minimale de F par $fils_{38,2}$.

chromosomes	**attributs de QoS**					F_i	$prob_i$	**EP**(P_2)
	TR_i	$securite_i$	$fiab_i$	$dispo_i$	$cout_i$			
$scomp_1$	3,42	0	0,06	0,005	3,91	1,73	0,124	1,74
$scomp_2$	3,59	0	0,05	0,001	4,21	1,81	0,130	
$scomp_3$	3,6	0	0,06	0,004	3,83	1,82	0,130	
$fils_2$	3,41	0	0,05	0,002	4,29	1,72	0,124	
$scomp_5$	3,16	0,33	0,04	0,003	3,7	1,66	0,119	
$scomp_6$	3,28	0	0,01	0,000	3,68	1,64	0,118	
$scomp_7$	3,3	0,33	0,01	0,001	3,76	1,72	0,124	
$fils_{38,2}$	3,64	0	0,03	0,001	3,97	1,83	0,131	
						13,92		

TABLE 3.8 – Scénario applicatif : les valeurs de QoS des services composites de la population P_2

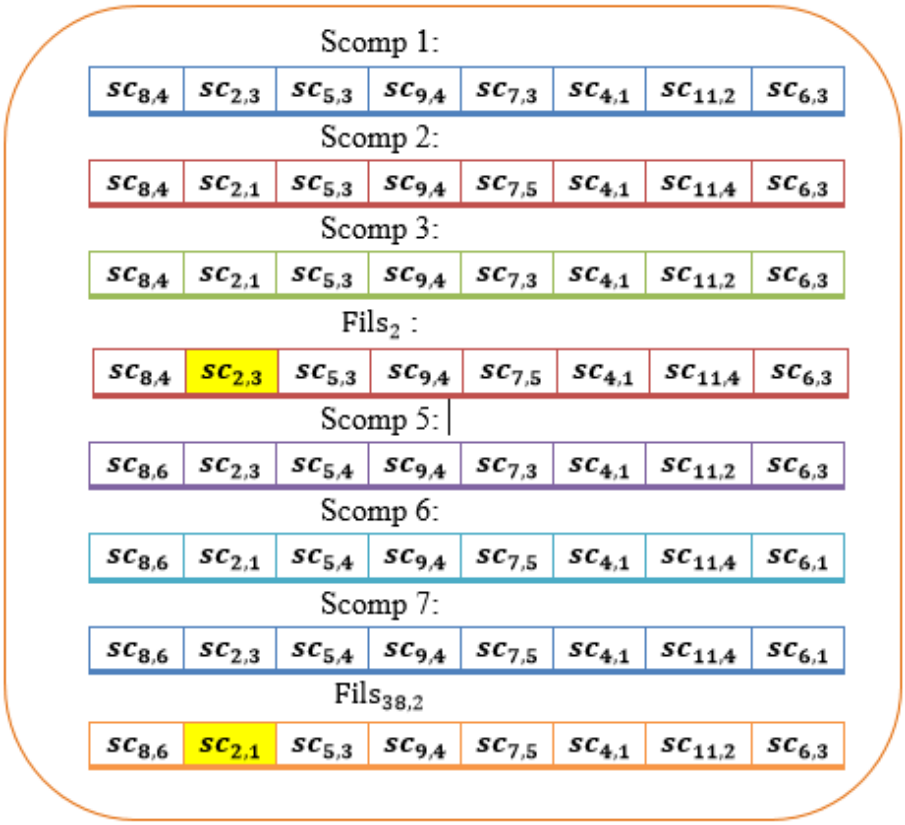


FIGURE 3.13 – Scénario applicatif : les services composites formant la population P_2

On incrémente le nombre de mutation réussie ($nbr_cR = 1$), et ($nbr_{gen} = 2$) > ($nbr_{gen_{max}}$) donc on arrête les opérations génétiques et on choisit la combinaison de services qui maximise la fonction d'utilité qui est le service composite $fils_{38,2}$.

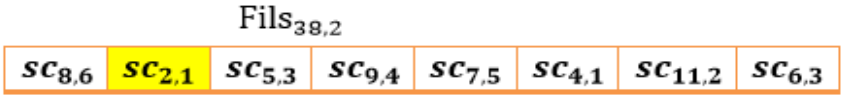


FIGURE 3.14 – Scénario applicatif : le service composite sélectioné

La figure 11 montre le plan concret obtenu de la sélection globale :

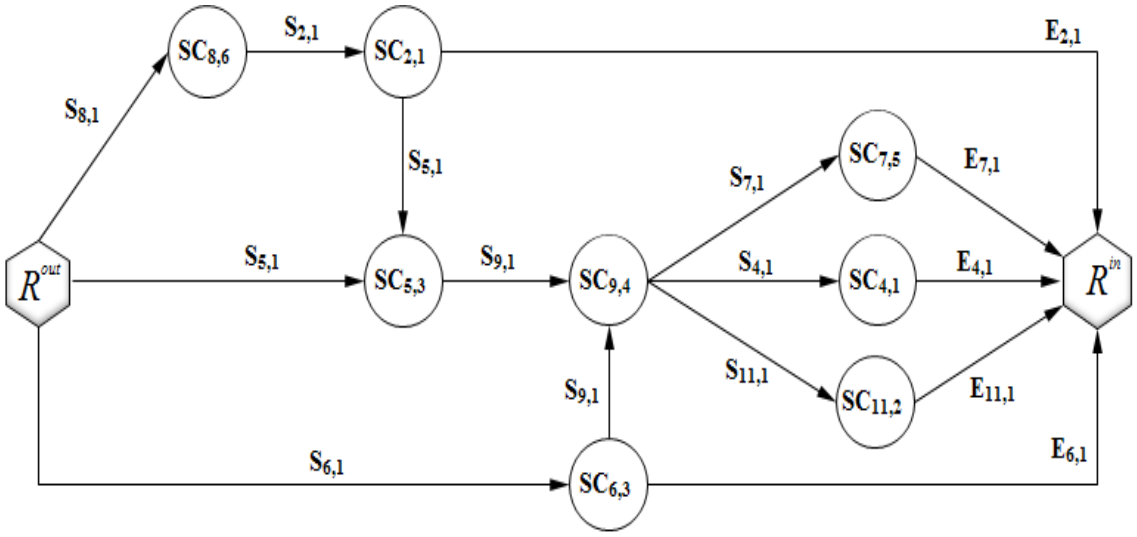
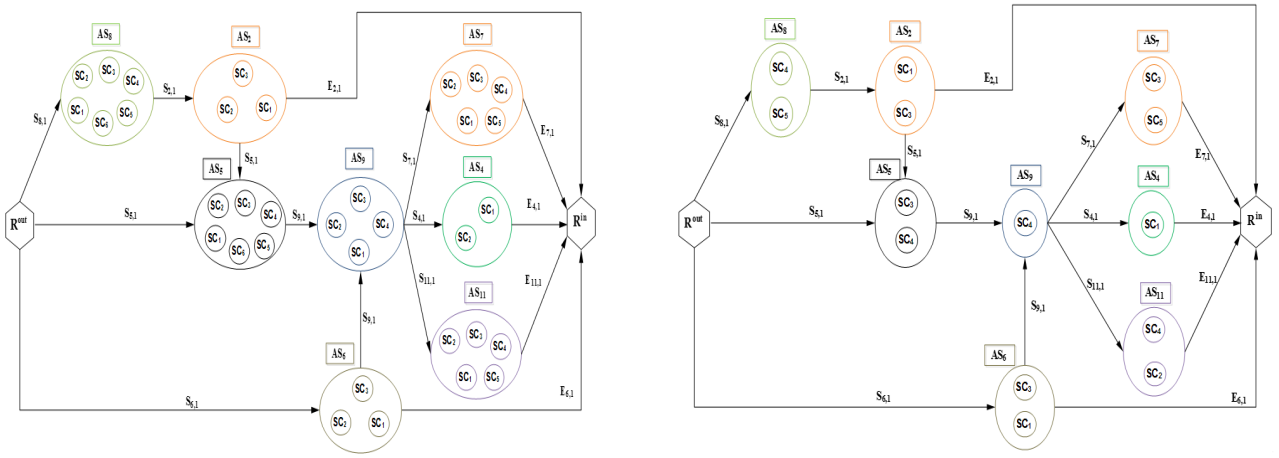


FIGURE 3.15 – Scénario applicatif : le plan concret obtenu de la sélection globale

Les étapes parcourues pour sélectionner la meilleur combinaison sont résumées ci dessous :

Etape1 :le plan optimal abstrait

Etape2 : le plan obtenu de la sélection locale



Etape3 : la population P_0

Scomp 1 : $sc_{8,4}$ $sc_{2,3}$ $sc_{5,3}$ $sc_{9,4}$ $sc_{7,3}$ $sc_{4,1}$ $sc_{11,2}$ $sc_{6,3}$

Scomp 2: $sc_{8,4}$ $sc_{2,1}$ $sc_{5,3}$ $sc_{9,4}$ $sc_{7,5}$ $sc_{4,1}$ $sc_{11,4}$ $sc_{6,3}$

Scomp 3: $sc_{8,4}$ $sc_{2,1}$ $sc_{5,3}$ $sc_{9,4}$ $sc_{7,3}$ $sc_{4,1}$ $sc_{11,2}$ $sc_{6,3}$

Scomp 4: $sc_{8,4}$ $sc_{2,3}$ $sc_{5,4}$ $sc_{9,4}$ $sc_{7,5}$ $sc_{4,1}$ $sc_{11,4}$ $sc_{6,1}$

Scomp 5: $sc_{8,6}$ $sc_{2,3}$ $sc_{5,4}$ $sc_{9,4}$ $sc_{7,3}$ $sc_{4,1}$ $sc_{11,2}$ $sc_{6,3}$

Scomp 6: $sc_{8,6}$ $sc_{2,1}$ $sc_{5,4}$ $sc_{9,4}$ $sc_{7,5}$ $sc_{4,1}$ $sc_{11,4}$ $sc_{6,1}$

Scomp 7: $sc_{8,6}$ $sc_{2,3}$ $sc_{5,4}$ $sc_{9,4}$ $sc_{7,5}$ $sc_{4,1}$ $sc_{11,4}$ $sc_{6,1}$

Scomp 8: $sc_{8,6}$ $sc_{2,3}$ $sc_{5,3}$ $sc_{9,4}$ $sc_{7,5}$ $sc_{4,1}$ $sc_{11,2}$ $sc_{6,3}$

Etape4 : la population P_1

Scomp 1 : $sc_{8,4}$ $sc_{2,3}$ $sc_{5,3}$ $sc_{9,4}$ $sc_{7,3}$ $sc_{4,1}$ $sc_{11,2}$ $sc_{6,3}$

Scomp 2: $sc_{8,4}$ $sc_{2,1}$ $sc_{5,3}$ $sc_{9,4}$ $sc_{7,5}$ $sc_{4,1}$ $sc_{11,4}$ $sc_{6,3}$

Scomp 3: $sc_{8,4}$ $sc_{2,1}$ $sc_{5,3}$ $sc_{9,4}$ $sc_{7,3}$ $sc_{4,1}$ $sc_{11,2}$ $sc_{6,3}$

Fils 2: $sc_{8,4}$ $sc_{2,3}$ $sc_{5,3}$ $sc_{9,4}$ $sc_{7,5}$ $sc_{4,1}$ $sc_{11,4}$ $sc_{6,3}$

Scomp 5: $sc_{8,6}$ $sc_{2,3}$ $sc_{5,4}$ $sc_{9,4}$ $sc_{7,3}$ $sc_{4,1}$ $sc_{11,2}$ $sc_{6,3}$

Scomp 6: $sc_{8,6}$ $sc_{2,1}$ $sc_{5,4}$ $sc_{9,4}$ $sc_{7,5}$ $sc_{4,1}$ $sc_{11,4}$ $sc_{6,1}$

Scomp 7: $sc_{8,6}$ $sc_{2,3}$ $sc_{5,4}$ $sc_{9,4}$ $sc_{7,5}$ $sc_{4,1}$ $sc_{11,4}$ $sc_{6,1}$

Scomp 8: $sc_{8,6}$ $sc_{2,3}$ $sc_{5,3}$ $sc_{9,4}$ $sc_{7,5}$ $sc_{4,1}$ $sc_{11,2}$ $sc_{6,3}$

Etape5 : la population P_2

Scomp 1: $sc_{8,4}$ $sc_{2,3}$ $sc_{5,3}$ $sc_{9,4}$ $sc_{7,3}$ $sc_{4,1}$ $sc_{11,2}$ $sc_{6,3}$

Scomp 2: $sc_{8,4}$ $sc_{2,1}$ $sc_{5,3}$ $sc_{9,4}$ $sc_{7,5}$ $sc_{4,1}$ $sc_{11,4}$ $sc_{6,3}$

Scomp 3: $sc_{8,4}$ $sc_{2,1}$ $sc_{5,3}$ $sc_{9,4}$ $sc_{7,3}$ $sc_{4,1}$ $sc_{11,2}$ $sc_{6,3}$

$Fils_2$: $sc_{8,4}$ $sc_{2,3}$ $sc_{5,3}$ $sc_{9,4}$ $sc_{7,5}$ $sc_{4,1}$ $sc_{11,4}$ $sc_{6,3}$

Scomp 5: $sc_{8,6}$ $sc_{2,3}$ $sc_{5,4}$ $sc_{9,4}$ $sc_{7,3}$ $sc_{4,1}$ $sc_{11,2}$ $sc_{6,3}$

Scomp 6: $sc_{8,6}$ $sc_{2,1}$ $sc_{5,4}$ $sc_{9,4}$ $sc_{7,5}$ $sc_{4,1}$ $sc_{11,4}$ $sc_{6,1}$

Scomp 7: $sc_{8,6}$ $sc_{2,3}$ $sc_{5,4}$ $sc_{9,4}$ $sc_{7,5}$ $sc_{4,1}$ $sc_{11,4}$ $sc_{6,1}$

$Fils_{38,2}$ $sc_{8,6}$ $sc_{2,1}$ $sc_{5,3}$ $sc_{9,4}$ $sc_{7,5}$ $sc_{4,1}$ $sc_{11,2}$ $sc_{6,3}$

Etape6 : le plan du service composite sélectionné

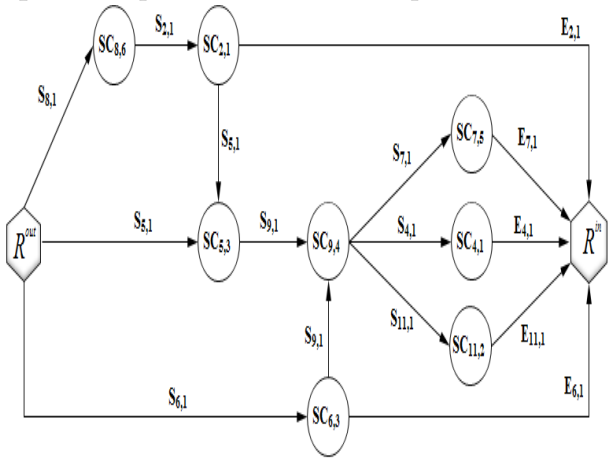


FIGURE 3.16 – Scénario applicatif : le plan cocnret de la composition

3.9 Conclusion

La saturation des environnements ubiquitaires par différents dispositifs, qui le rend un environnement intelligent rend aussi la composition de service un domaine de recherhe très vaste. Dans le présent chapitre, nous avons présenté notre proposition, nomée QSCGA, qui consiste en une approche de composition de services sensible au contexte avec QoS basée sur les algorithmes génétiques. QSCGA répond aux exigences locales et globales de l'utilisateur .

4

Simulation et évaluation de performances

4.1 Introduction

Nous venons de détailler dans le chapitre précédent l'approche proposée pour le problème de composition de services. Le présent chapitre est dédié à la validation par simulation de notre contribution. En effet, nous décrivons dans un premier temps les outils logiciels utilisés (Eclipse et JAVA) lors de l'implémentation, puis nous évaluons les performances de l'approche en comparant ses résultats avec ceux de la solution proposée par Ben Mabrouk et al, [34].

4.2 Présentation des outils utilisés

4.2.1 Environnement de travail

La validation de l'approche proposée a été réalisée sur une machine dont les caractéristiques sont :

- Processeur : Intel Core(TM) i3, avec une vitesse de 1.80 GHZ,
- Capacité de la RAM : 4 Go,
- Systéme d'exploitation : Windows 8,
- Langage de développement : JAVA.

4.2.2 Le langage JAVA

Pour le langage de programmation notre choix s'est porté sur le langage JAVA, car il est un langage orienté objet et simple, ce qui réduit les risques d'incohérence. Il est portable, c.à.d, il peut être utilisé sous Windows, Linux, Macintosh et d'autres plateformes sans aucune modification. Enfin, il possède une riche bibliothèque de classes comprenant de diverses fonctions.

4.2.3 Eclipse

Eclipse possède de nombreux points forts qui sont à l'origine de son énorme succès dont les principaux sont :

- Une plateforme ouverte pour le développement d'applications et extensible grâce à un mécanisme de plug-in ;
- Plusieurs versions d'un même plug-in peuvent cohabiter sur une même plateforme ;
- Support de plusieurs plates-formes d'exécution : Windows, Linux, Mac OS, etc.

4.3 Description du système

4.3.1 Les structures de données

Dans le contexte des algorithmes génétiques, on utilise les termes population, individu, chromosome et gène, tels qu'une population est constituée d'un ensemble de chromosomes et un chromosome est une suite de gènes.

Les algorithmes génétiques utilisent des opérations de mutation et de croisement. Dans cette section, nous décrivons les structures de données les plus importantes que nous utilisons pour représenter les paramètres génétiques de notre implémentation.

- Gène : est équivalent à un service concret (noté SC) de l'environnement dont les valeurs de paramètres de QoS sont générés aléatoirement.

- Chromosome : est équivalent à un service composite (noté SComp) représenté par une classe paramétrée par :
  1) le vecteur de services concrets qui forment le service composite et
  2) les paramètres de qualité du service composite (QoSComp).

– Population : est équivalente à une collection de services composites, appelée aussi génération.

4.3.2 Structuration de l'algorithme de composition

L'implémentation en JAVA de l'approche génétique pour la composition nécessite la mise en œuvre de trois types de classes (une classe principale, un ensemble de classes pour la sélection locale et un ensemble de classes pour les opérations génétiques). Ces classes se divisent en trois packages.

4.3.2.1 Le package de sélection locale

Les classes de ce packages (sélection et mise à l'échelle) décrivent le processus de la sélection locale des services concrets.

– ***Mise_à_echelle :*** le rôle de cette classe est de transformer les valeurs de différents paramètres de QoS en valeurs comprises entre 0 et 1 (Sec. 3.5).

– ***SelectionLocale :*** cette classe modifie le plan optimale en supprimant les services concrets dont la QoE est inférieur à un seuil spécifié par l'utilisateur.

4.3.2.2 Le package des opérations génétiques

Il contient les classes de mise en œuvre des opérations génétiques (mutation, croisement, etc.) pour la sélection du meilleur service composite.

– ***Initialisation :*** cette classe permet d'initialiser la population. Elle sélectionne aléatoirement, en utilisant une fonction mathématique prédéfinie (Random), des services concrets dans chaque service abstrait pour participer à la composition.

– ***Sélection :*** cette classe se charge de la sélection de deux parents ($parent_1$ et $parent_2$) pour l'opération de croisement.

– ***Fitness :*** elle est responsable de calculer la valeur d'évaluation d'un individu.

– ***Croisement :*** une classe qui prend comme entrée un couple d'individus déterminé par l'opération de sélection et retourne le fils (résultat du croisement).

– ***Remplacement :*** une classe qui décide de la position pour insérer le nouvel individu (service composite) après le retrait d'un ancien.

4.3.2.3 Le package de la classe principale

Ce package contient une seule classe qui est le code Java de l'algorithme de composition proposé dans le chapitre 3. Cette classe reçoit les données d'entré nécessaires pour paramétrer l'application.

4.4 Evaluation de performances

Pour cela, nous avons mesuré l'évolution du temps de séléction engendré par l'algorithme de composition de services proposé (CSAG) en fonction du nombre de services. En effet, nous avons effectué deux tests, le premier est fait en fixant le nombre de services abstraits à 50 tout en variant le nombre de services concrets dans chaque classe de services abstraits de 10 à 50. Le deuxième test est effectué en fixant le nombre de services concrets pour chaque service abstrait à 50 et en variant le nombre de services abstraits de 10 à 50. Les résultats obtenus sont comparés avec ceux de l'approche proposée dans [34].

4.4.1 Variation du nombre de services concrets

Dans ce test, nous mesurons le temps de sélection en fonction de la variation du nombre de services concrets (de 10 à 50). Le nombre de services abstraits étant fixé à 50. Les résultats obtenus sont la moyenne de 10 exécutions, ils sont représentés sur la figure 1.

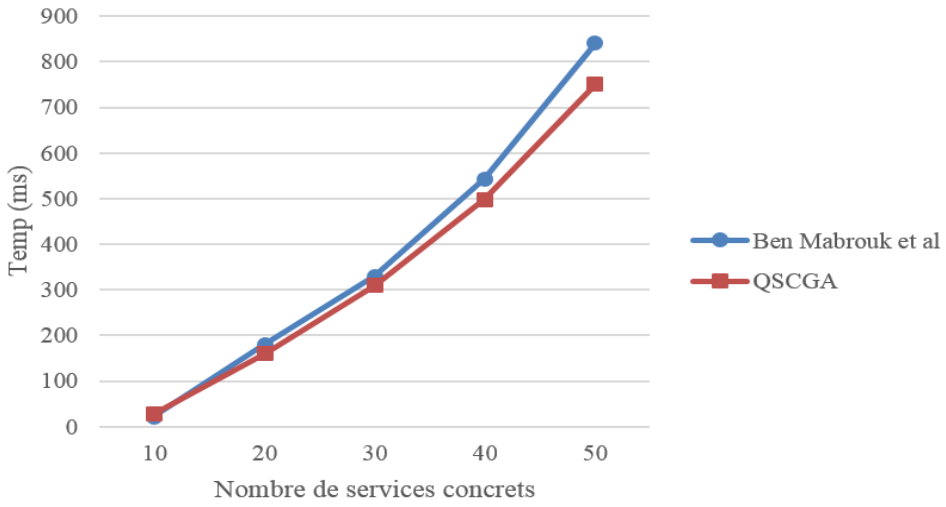


FIGURE 4.1 – Temps de sélection en fonction du nombre de services concrets pour 50 services abstraits

La figure montre que le temps de sélection globale engendré par l'algorithme de composition proposé (QSCGA) est nettement meilleur en le comparant à celui de [34]. Ceci est dû au fait que QSCGA sélectionne un sous ensemble de combinaisons parmi celles qui sont possibles afin d'appliquer les opérations génétiques (sélection, mutation, croisement et remplacement) sur les meilleurs services composites de la population. Tandis que l'approche de Ben Mabrouk et al. [34] prend en considération toutes les combinaisons possibles.

On remarque qu'au début, l'approche proposé dans [34] est meilleure que l'algorithme QSCGA. Ensuite, plus le nombre de services concrets augmente plus QSCGA montre une meilleure performence et l'écart en temps de sélection entre les deux algorithmes augmente. Ce qui nous permet d'affirmer que notre contribution QSCGA est plus scalable que celle de Ben Mabrouk et al, [34].

Ces résultats montrent que notre méthode pour la composition globale de services est stable et conduit à un temps de calcul raisonnable. En effet, pour 50 services abstraits et 50 services concrets dans chaque classe de services abstraits, le temps de calcul est inférieur à une seconde (760 millisecondes).

4.4.2 Variation du nombre de services abstraits

Dans ce test, nous mesurons le temps de sélection en fonction de la variation du nombre de services abstrait (de 10 à 50). Le nombre de services concrets dans chaque classe de services abstraits étant fixé à 50. La figure 2 montre les résultats obtenus après 10 exécutions.

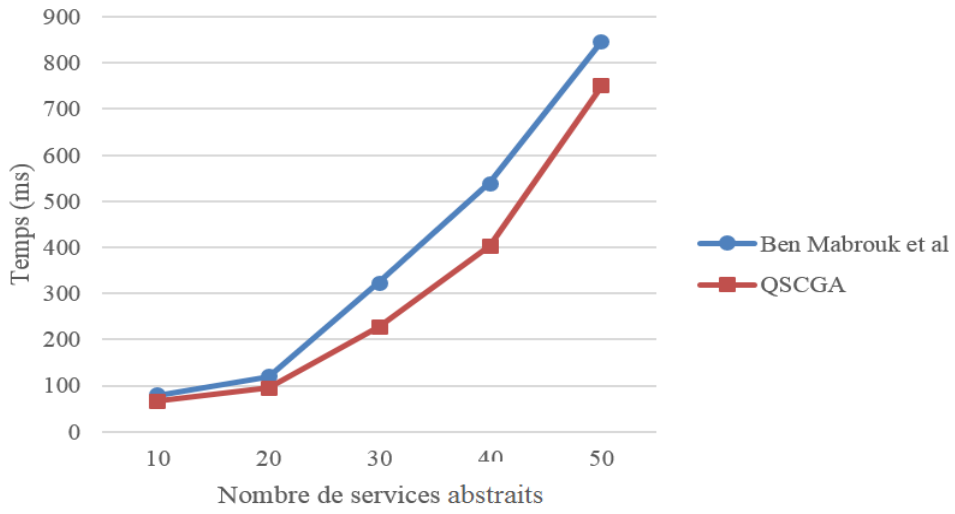


FIGURE 4.2 – Temps de sélection en fonction du nombre de services abstraits pour 50 services concrets.

La figure montre qu'au début les deux approches sont presque équivalentes en terme de temps de sélection. Eensuite l'écart entre l'approche QSCGA et celle de Ben Mabrouk et al, [34] croît avec l'augmentation du nombre de services abstraits. Bien que le temps de sélection engendré par notre approche augmente, en raison de sa dépendance du nombre de services abstraits, ces prformaces restent toujours meilleures que celles de [34].

Les résultats montrent que QSCGA est meilleur que l'algorithme [34]. Ce gain peut atteindre les 90 ms pour la première série de tests et 115 ms pour la deuxième. En effet, cet apport est dû à la sélection locale qui réduit le nombre de services concrets entrainant l'application de l'algorithme génétiques sur une population de petite taille ce qui réduit le temps de la sélection.

4.5 Conclusion

Au cours de ce chapitre, nous avons réalisé une étude comparative de notre contribution baptisée QSCGA avec un des algorithmes proposés dans la littérature et ce à travers la simulation. Les résultats montrent que l'algorithme proposé améliore le temps de sélection, cela est dû principalement à la phase de sélection locale des services concrets et l'utilisation des algorithmes génétiques sur le plan globale. En outre, la solution proposée est proche de l'optimum en termes de QoS dans un temps d'exécution acceptable.

Conclusion générale et perspectives

L'émergence des réseaux dynamiques, c'est-à-dire des environnements constitués d'équipements hétérogènes et mobiles amène à repenser les besoins des fonctionnalités complexes (c.à.d, des fonctionnalités qu'un seul service atomique ne peut fournir). L'omniprésence des équipements informatiques fait émerger en effet un nouveau besoin, celui de l'ubiquité des services offerts par les applications. Cependant, l'accès aux fonctionnalités d'un même service depuis n'importe quel équipement qui entoure l'usager, reste encore difficile à concevoir et à mettre en œuvre dans des environnements dynamiques. Dans de tels environnements, les dipositifs sont mobiles et aucune hypothèse sur la disponibilité de services requis ne peut être faite. La connectivité entre les différents dispositifs fluctue et ne peut donc être garantie.

Dans ce travail, nous avons proposé une approche, baptisée QSCGA (pour QoS-aware Service Composition based Genetic Algorithm), de sélection globale de services dont le principe est inspiré des algorithmes génétiques. Cette approche permet d'une part, d'efectuer une sélection locale tenant en compte les préférences et les exigences locales de l'utilisateur et d'autre part, de garantir le respect des exigences globales en choisissant le service composite optimal en termes de qualité de service. L'algorithme QSCGA permet aussi une sélection dynamique de services en prenant en considération la nature dynamique des environnements ubiquitaires, telles que l'apparition et la disparaissions de services pouvant se produire lors de l'invocation. La sélection utilise, par ailleurs, les informations contextuelles à savoir la localisation des entités (dispositifs et utilisateurs) et le niveau d'énergie des équipements fournissant les services.

Afn de valider l'approche proposée, nous avons développé un simulateur écrit en langage JAVA. Les séries de tests réalisés et l'analyse comparative des résultats obtenus par l'algorithme QSCGA avec ceux de [34], montre une meilleure performance en terme de scalabilité et de temps de sélection au profit de notre contribution. En effet, cet apport est justifié par la méthode de sélection adoptée et le choix judicieux des paramètres

d'initialisation de l'algorithme génétique.

En quise de perspectivess :

1. Le plan abstrait global et optimal sont supposés existants, il serait important de décrire la façon d'obtenir ce plan global abstrait et l'optimiser ;
2. Fixer des critères de sélection pour les services composites qui forment la population initiale au lieu de les sélectionner aléatoirement ;
3. Prendre en considération d'autres attributs de contexte à part l'énergie et la localisation, à savoir le contexte de l'environnement (température, humidité, etc.), le temps (jour et nuit), et d'autres.
4. Combiner l'algorithme génétique avec d'autres techniques d'optimisation, par exemple les algorithmes de colonies de fourmis.

Bibliographie

[1] : M. Weiser. The Computer for the Twenty-First Century. *Scientific American Journal*, 265(3) : pages 94-104, 1991.

[2] : A. Kouicem. *Composition dynamique de services en environnement ubiquitaire*, Mémoire de magister en informatique, université A.Mira, Bejaia, 2008.

[3] : B. Cogrel, B. Daachi, Y. Amirat and A. Chibani. Sélection de services basée sur l'impact en environnement ubiquitaire, In *Proceedings of the 7th French-speaking Conference on Mobility and Ubiquity Computing*, pages 9-15, 2011.

[4] : D. E. Dedefa. *Services Pervasifs Contextualisés : Modélisation et Mise en OEuvre*. Thèse de doctorat en informatique, Institut national des sciences appliquées, Lyon, 2007.

[5] : F. Laforest and F. Le Mouël. Systèmes d'information pervasifs, Cours de master, Grenoble, 2006.

[6] : R. Ben Halima. *Conception, implantation et expérimentation d'une architecture en bus pour l'autoréparation des applications distribuées à base de services web*. Thèse de doctorat en informatique, Université de Toulouse et l'université de Sfax, France, 2009.

[7] : A. Yachir. *Composition dynamique de services sensibles au contexte dans les systemes intelligents ambiants*. Thèse de doctorat en Informatique, Université des sciences et de la technologie Houari Boumediène (USTHB), 2014.

[8] : Y. D. Boromberg. *Résolution de l'hétérogénéité des intergiciels d'un environnement ubiquitaire*. Thèse de doctorat en informatique, Université de Versailles-Saint Quentin en Yvelines, 2006.

[9] : W. Emmerich. Software Engineering and Middleware : A Roadmap. In *Proceedings of the Conference on The future of Software engineering*, pages 117-129, 2000.

[10] : A. Bottaro. *Composition contextuelle de services dans les réseaux d'équipements pervasifs*. thèse de doctorat en informatique, Université Joseph Fourier, Grenoble, 2008.

[11] : W. Jouve, *Approche déclarative pour la génération de canevas logiciels dédiés à l'informatique ubiquitaire*. Thèse de doctorat en Informatique, Université de Bordeaux 1, 2009.

[12] : J. Mercadal. *Approche langage au développement logiciel : application au domaine des systèmes d'informatique ubiquitaire*. Thèse de doctorat en informatique, Université de Bordeaux, 2011.

[13] : D. Hoareau. *Composants ubiquitaires pour réseaux dynamiques*. Thèse de doctorat en informatique, Université de Bretagne Sud, Décembre 2007.

[14] : A. T. Manes. Service-Oriented Architecture : Developing the Enterprise Roadmap. Volume 2, pages 22, 2006.

[15] : B. Schilit and M. Theimer. Disseminating Active Map Information to Mobile Hosts, *IEEE Network Journal*, 8(5) : pages 22–32, 1994.

[16] : P. J. Brown, J. D. Bovey and X. Chen. Context-aware Applications : from the Laboratory to the Marketplace, *IEEE Personal Communications Journal*, 4(5) : pages 58–64, 1997.

[17] : T. Chaari. *Adaptation d'applications pervasives dans des environnements multi-contextes*, Thèse de doctorat en informatique, Institut national des sciences appliquées de Lyon, France, 2007.

[18] : G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, , and P. Steggles. Towards a Better Understanding of Context and Context-Awareness, *In Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pages 304–307, 1999.

[19] : P. Brezillon, M. R. Borges, J. A. Pino and J. Ch. Pomerol. Context-awareness in group work : three case studies. In *Proceedings of the international Conference on Decision Support Systems (DSS 2004)*, pages 115-124, 2004.

[20] : M. A. Razzaque, S. Dobson and P. Nixon. Categorization and modeling of quality in contextinformation. Workshop on AI and Autonomic Communications, held at International Joint Conference on Artificial Intelligence, 2005.

[21] : 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 de Québec, 2009.

[22] : M. P. Papazoglou. *Service-oriented computing : Concepts, characteristics and directions. In proceedings of the 4th International Conference on Web Information Systems Engineering (WISE'03)*, IEEE, pages 3-12, 2003.

[23] : C. Marin. *Une approche orientée domaine pour la composition de services*. Thèse de doctorat en informatique, Université Joseph Fourier, Grenoble I, 2008.

[24] : S. Chollet. *Orchestration de services hétérogènes et sécurisés*. Thèse de doctorat en informatique, Université Joseph Fourier, Grenoble I, 2009.

[25] : F. Baligand. *une Approche Déclarative pour la Gestion de la Qualité de Service dans les Compositions de Services*. Thèse de doctorat en Informatique Spécialité "Informatique temps réel, robotique et automatique", Ecole Nationale Supérieure des Mines de Paris, 2008.

[26] : K. Boukadi. *Coopération interentreprises à la demande : Une approche flexible à base de services adaptables*. Thèse de doctorat en Informatique, Ecole Nationale Supérieure des Mines de Saint-Etienne, 2009.

[27] : A. Hock-Koon : *Contribution à la compréhension et à la modélisation de la composition et du couplage faible de services dans les architectures orientées services*, Thèse de doctorat en informatique, Université de Nantes, 2013.

[28] : C. Lopez-Velasco. *Sélection et composition de services Web pour la génération d'applications adaptées au contexte d'utilisation*. Thèse de doctorat en Mathématiques, Sciences et Technologie de l'Information (spécialité Informatique), Université Joseph-Fourier-Grenoble I, 2008.

[29] : B. Benatallah, Q. Z. Sheng, and M. Dumas. The Self-Serv environment for Web services composition. *Internet Computing Journal, IEEE*, 7(1) : pages 40-84, 2003.

[30] : N. BELAID. *Modélisation de services et de workflows sémantiques à base d'ontologies de services et d'indexations. Application à la modélisation géologique*. Thèse de doctorat en informatique et application, Ecole Nationale Supérieure de Mécanique et d'Aérotechnique, France, 2011.

[31] : A. Bekkouche. *Composition des Services Web Sémantiques A base d'Algorithmes Génétiques*. Mémoire de magistère en informatique, Université Abou-bekr Belkaid Tlemcen, Algérie, 2012.

[32] : B. Orriëns, J. Yang, and M. P. Papazoglou. Service Component : a mechanism for web service composition reuse and specialization. *Journal of Integrated Design and Process Science*, 8(2) : pages 13-28, 2004.

[33] : I. BRAKNI. *Planification multi-agents pour la composition dynamique*. Mémoire Ingéniorat en informatique, Université de Tébessa, 2010.

[34] : N. Ben Mabrouk, S. Beauche, E. Kuznetsova, N. Georgantas, and V. Issarny. QoS-aware Service Composition in Dynamic Service Oriented Environments. Middleware 2009, Springer Berlin Heidelberg, pages 123-142, 2009.

[35] : F. Rosenberg, P. Leitner, A. Michlmayr, P. Celikovic and S. Dustdar. Towards Composition as a Service – A Quality of Service Driven Approach, In Proceedings of Distributed Systems Group. *In Proceedings of the 25th International Conference on Data Engineering,*, Pages 1733-1740, 2009.

[36] : M. Alrifai, T. Risse. Combining Global Optimizationwith Local Selection for Efficient QoS-aware Service Composition. *In Proceedings of the 18th international conference on World wide web*, Pages 881-890, 2009.

[37] : K. Tari, Y. Amirat, A. Chibani, A. Yachir, and A. Mellouk. Context-aware Dynamic Service Composition in Ubiquitous Environment. In *Proceedings of International conference on Communications (ICC)*, pages 1-6, 2010.

[38] : L. Zhao, Y. Ren, M. Li and K. Sakurai. Flexible service selection with user-specific QoS support in service-oriented architecture. Elsivier Network Journal and Computer Applications, 35(3) : 962–973, 2011.

[39] : M. Henrique Cruz Torres, T. Holvoet. *Composite service adaptation : A QoS-driven approach. In Proceedings of the 5th International Conference on Communication System Software and Middleware (ACM)*, page 8, 2011.

[40] : A. Yachir, Y. Amirat, A. Chibani, and N. Badache.Towards an event-aware approach for ubiquitous computing based on automatic service composition and selection. *annals of telecommunications-annales des télécommunications*, 67(7-8) : pages 341-353, 2012.

[41] : T. Gong, Z. Hu, H. Liu, F. Lin, D. Zhou and H. Tian. A Context-aware Computing Mediated Dynamic Service Composition and Reconfiguration for Ubiquitous Environment. *In Proceedings of the 3rd international conference on internet of things*, pages 16-23, 2012.

[42] : D. Efstathiou, P. McBurney, S. Zschaler, and J. Bourcier. Flexible QoS-Aware Service Composition in Highly Heterogeneous and Dynamic Service-Based Systems. *In Proceedings of the 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 592-599, 2013.

[43] : C. Lopez-Velasco. *Sélection et composition de services Web pour la génération d'applications adaptées au contexte d'utilisation.* Thèse de doctorat en mathématiques, sciences et technologie de l'information, Université Joseph Fourier de Grenoble, 2008.

[44] : S. Achiri. *Une architecture agents pour l'adaptation au contexte des systèmes d'information ubiquitaires*. Mémoire de magister en informatique, Université Badji Mokhtar, Annaba, 2012.

[45] : A. Kansal and F. Zhao. Fine-grained energy pro-ling for power-aware application design. *ACM SIGMETRICS Performance Evaluation Review*, 36(2) : pages 26-31, 2008.

[46] :N. DURAND. *Algorithmes génétiques et autres outils d'optimisation appliqués à la gestion de trafic aérien*. Thèse d'Habilitation à Diriger des Recherches (HDR), Institut Polytechnique de Toulouse, 2004.

[47] : T. Vallee and M. Yildizoglu. Présentation des algorithmes génétiques et de leurs applications en économie, *Revue d'économie politique*, 114(6) : pages 711-745. Septembre 2004.

[48] : R. Sabourin. *Introduction aux Algorithmes Génétiques*, Cour de Dr Sbourin, École de technologie supérieure, Montréal.

[49] : D. Goldberg. *GAs in Search, Optimization, and Machine Learning*. Edition Addison Wesley Publishing Company, 1989.