# CSC (Composant, Service, Connecteur) - Une plateforme d'adaptation pour applications multimédia

Derdour Makhlouf, Philippe Roose, Marc Dalmau, Nacéra Ghoualmi-Zine

# CSC (Composant, Service, Connecteur)

## Une plateforme d'adaptation pour applications multimédia

Derdour Makhlouf[1], Philippe Roose[2], Marc Dalmau[2], Nacéra Ghoualmi-zine[1]

[1]Département d'Informatique
Université Badji Mokhtar - Annaba
Annaba, Algérie
{m.derdour, ghoualmi}@yahoo.fr

[2]LIUPPA
Université de Pau et des Pays de l'Adour
Anglet, France
{Philippe.Roose, Marc.Dalmau}@iutbayonne.univ-pau.fr

*Résumé*— **La tendance vers les services ubiquitaires et le tout multimédia, la multiplication des terminaux mobiles et la généralisation des réseaux sans fil impliquent des changements dans la conception et l'exécution des applications logicielles. Les systèmes ubiquitaires sont des systèmes dynamiques qui modifient leur comportement en fonction des besoins de l'utilisateur et des capacités matérielles lors de l'exécution en se basant sur des informations contextuelles. Comme il n'est pas souhaitable de développer ces systèmes à partir de zéro à chaque fois, une architecture logicielle spécifique offrant des possibilités d'adaptations dynamiques des systèmes est nécessaire. Elle doit être capable de créer des adaptations à l'exécution afin d'offrir un comportement dynamique et adaptatif pour les utilisateurs. Dans cet article nous présentons une plateforme d'adaptation supervisée pour les applications basées composants.**

**La plateforme CSC (composant, service, connecteur) est basée sur un modèle composant/service qui permet une auto-adaptation des applications basées composants en se basant sur une architecture orienté services pour la fourniture des services d'adaptation qui seront intégrés dans des connecteurs d'adaptation.**

***Mots-clés- composant ; connecteur; architecture logicielle; multimédia;adaptation.***

## I. INTRODUCTION

Dans le cadre de l'informatique ubiquitaire, l'environnement d'exécution d'une application est constitué de machines hétérogènes (*PC, PDA, Smartphone, etc.*) en ressources matérielles (*taille écran, modes d'interactions, mémoire, batterie, interfaces réseaux, etc.*) appartenant à des utilisateurs avec des besoins différents et manipulant des médias de différents types (*vidéo, son, image, texte*). Ces caractéristiques imposent de structurer l'application en une organisation d'entités logicielles relativement indépendantes qui coopèrent et interagissent afin de faciliter son adaptation au contexte d'utilisation.

Malheureusement cette organisation structurelle et comportementale ne permet pas de résoudre tous les problèmes d'hétérogénéité. Au-delà du respect des contraintes fonctionnelles, le remplacement d'un composant par un autre réclame la satisfaction de plusieurs conditions et la vérification de plusieurs propriétés (*homogénéité des composants, cohérence de l'assemblage, stabilité de l'application, traçabilité des choix d'adaptation, etc.*). Cela demande une réflexion sur la conception et notamment sur la configuration des applications. Cette réflexion consiste à séparer les aspects fonctionnels de ceux de l'adaptation et de prévoir des mécanismes dynamiques et reconfigurables durant tout le cycle de vie de l'application.

Le développement d'un système d'adaptation sensible aux caractéristiques fonctionnelles et non-fonctionnelles nécessite de répondre à deux questions : comment concevoir une plateforme garantissant une adaptation dynamique sensible au contexte ? Comment concevoir des adaptateurs pour assurer les adaptations nécessaires? Les aspects importants pour la conception d'une solution d'adaptation du contenu sont la diversité du contenu, la description de l'environnement, la gestion du contexte et l'adaptation. Quant à l'hétérogénéité des environnements, on la rencontre à plusieurs niveaux : l'environnement de l'utilisateur (bruyant, sombre, géographique, etc.), du terminal (*taille écran, codecs supportés, etc.*) et des composants de l'application (*interfaces de communication de type client-serveur, RPC, etc.*). Pour cela une conception et une implémentation d'un tel système impliquent des efforts dans plusieurs domaines.

Il existe deux grands axes pour l'adaptation des applications basées composants, celui de l'adaptation fonctionnelle comme MUSIC [19] et MADcAR [20] basée sur le réassemblage ou la reconfiguration, et celui de l'adaptation non-fonctionnelle comme SCL [21] basée sur le comportement des composants. Dans notre approche nous avons travaillé sur la partie non-fonctionnelle pour l'adaptation des flux échangés entre les composants afin de traiter le problème d'hétérogénéité. On ne cherche pas la modification ou le remplacement de fonctionnalités, mais on adapte ces dernières au contexte d'exécution. Ceci peut se traduire par des adaptations d'assemblages (appel de services différents, redéploiements, remplacement de composants/services, etc.) se traduisant par la conservation de la même fonctionnalité mais avec une qualité de service différente.

L'introduction massive de données multimédia dans les systèmes ubiquitaires/pervasives conduit à manipuler différents types de médias, ce qui provoque l'apparition de plusieurs problèmes influençant l'interopérabilité des composants tels

que l'hétérogénéité des flux de données échangés entre ses composants. Ces problèmes sont liés à la taille des données (*les flux vidéo sont difficilement gérable selon le type de connexion*), les encodages des données (*formats, codecs, conteneurs, qualité d'encodage*), modalité (*texte reçu alors que la personne est malvoyante, etc.*). Les services d'adaptation sont une solution pour résoudre ce problème qui représente l'un des défis majeur de ces applications. Ils proposent des mécanismes permettant d'assurer l'adaptation des flux de données multimédia échangés entre des composants hétérogènes. Dans cette perspective, CSC propose une plateforme d'exécution et d'adaptation des applications multimédia.

L'entité principale dans notre proposition est le connecteur (ne possédant pas de caractère fonctionnel), qui propose des solutions pour répondre aux problèmes d'adaptation pour les composants métiers et permet la résolution des hétérogénéités des données sans modifier les fonctionnalités d'un tel système. Il est représenté par un composant de première classe. Il est de première classe car il ne se contente pas d'avoir le rôle traditionnel lié à la communication mais est également en charge de l'adaptation (de façon unitaire ou par assemblage de connecteurs) des données. Ce type de connecteur est de plus reconfigurable et capable d'adapter les flux de données multimédia selon la situation.

## II. MOTIVATION

Notre principale motivation est de proposer une plateforme pour maintenir la cohérence des configurations réalisées par assemblages de composants hétérogènes (*flux multimédia, interfaces de communication, etc.*) en utilisant MMSA [1] qui fournit des nouveaux types d'interfaces graphique et de connecteurs dotés d'une sémantique plus riche. L'utilisation de ces interfaces permet la détection automatique des points d'hétérogénéité entre composants, alors que l'utilisation des connecteurs d'adaptation permet la résolution de ces hétérogénéités.

Dans la plupart des plateformes d'adaptation et d'auto-adaptation [19, 20, 21] on trouve que:

*a) La gestion des assemblages ne prend pas en considération les hétérogénéités comportementales des composants de l'architecture logicielle.*

*b) Peu de plateformes permettent de définir de nouveaux connecteurs avec des traitements ad'hoc qui assurent des préoccupations non-fonctionnelles des composants (adaptation, sécurité, communication, conversion, etc.).*

*c) Il n'y a pas de correspondance directe et automatique entre les architectures (les modèles) et les applications conçues suivant ces architectures (instances).*

Afin de répondre à ces manques, nous proposons CSC, une plateforme d'auto-adaptation basée sur MMSA pour décrire les architectures des applications à vocation multimédia et fournir les besoins d'adaptation nécessaires. La plateforme forme est basée services et offre une architecture à trois couches particulièrement adaptée à l'adaptation des flux multimédia (*types, formats, propriétés*) permettant de résoudre les problèmes d'hétérogénéité des composants.

## III. APPROCHE MMSA

La mise en évidence des incompatibilités des flux de données échangés entre composants est une nécessité dans de telles approches basée composants. En effet, les architectures logicielles valident les aspects fonctionnels, ce qui n'est pas suffisant pour garantir un assemblage réaliste et remédier aux problèmes d'hétérogénéité des flux de données échangés. Pour mettre en évidence ces incompatibilités et permettre de trouver des solutions, une approche basée modèle appelée MMSA (*Meta-model MultiMedia Software Architecture*) a été proposé dans [1]. Elle permet la description d'architectures logicielles à l'aide d'un profil UML dédié exprimant un système logiciel comme une collection de composants qui manipulent différents types et formats de données et qui interagissent entre eux par l'intermédiaire de connecteurs incluant les connecteurs d'adaptation. MMSA fournit une configuration cohérente qui pourra être utilisé pour la configuration et l'exécution de l'application. Néanmoins, le contrôle et le suivie de l'application est nécessaire afin de répondre au éventuelle changement dynamique du contexte et conservé la cohérence de l'application.

MMSA est un méta modèle générique pour la description des architectures logicielles tout en intégrant des concepts liés aux données multimédia et la qualité de service. Ceci lui a permis de présenter de façon séparée les paramètres de flux et de média en ce sens qu'ils présentent un aspect très important des configurations et des assemblages des composants. Les points forts de MMSA sont la prise en compte de l'aspect multimédia et la séparation entre l'aspect fonctionnel et non-fonctionnel des composants.

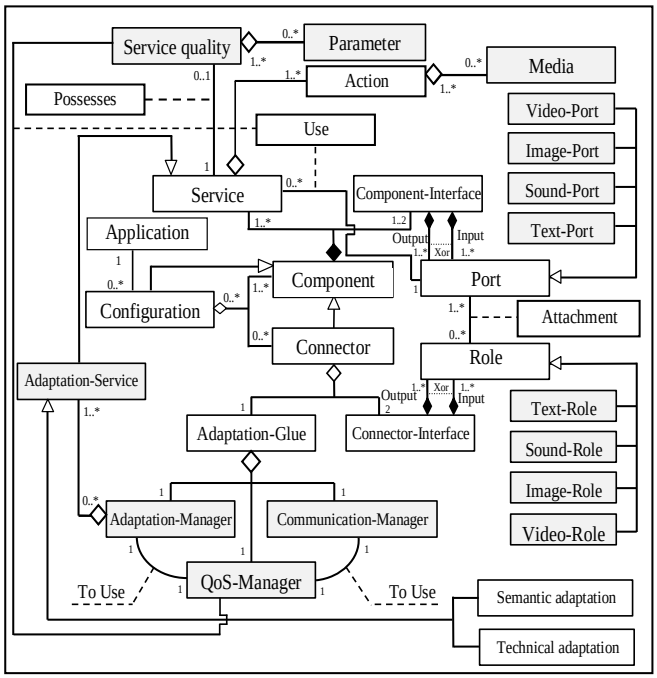



Figure 1. Diagramme de classe du méta-modèle MMSA

MMSA est un méta-modèle d'architecture logicielle pour applications multimédia intégrant les propriétés des flux de données multimédia. L'adaptation des flux de données est déportée sur des connecteurs appelés connecteurs d'adaptation. Ces derniers intègrent les services d'adaptation nécessaires ainsi que des extensions qualitatives de ces services afin d'offrir une mesure de QdS reflétant l'évolution du flux de données suite aux adaptations.

Les connecteurs servent à relier et à faire communiquer des composants entre eux, y compris lorsque l'interface réseau est différente. Un connecteur est composé d'une glu et deux interfaces (requise et fournie). Les types de connecteurs d'adaptation ont été proposés dans [2] pour assurer l'adaptation des flux multimédia échangés entre les composants de l'application.

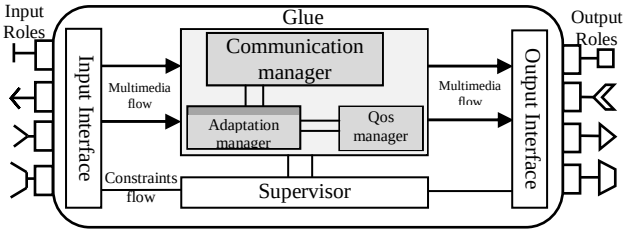



Figure 2. Modèle de connecteur d'adaptation

Permettre à des composants hétérogènes d'interagir est une tâche non négligeable. L'adaptation étant considérée comme un besoin non-fonctionnel d'un composant, cette tâche doit être assurée par un autre élément, le connecteur. Il fournit les aspects non-fonctionnels (*communication, adaptation, sécurité, etc.*) dont le composant a besoin. Le rôle du connecteur d'adaptation est de recevoir les données, de les adapter selon les directives du gestionnaire de QoS (*qui permet de modifier les paramètres des services d'adaptation afin de gérer la qualité fournie par ces derniers en fonction des besoins*) et de les acheminer vers le composant ou le connecteur destinataire selon le format et la forme souhaitée par ce dernier. Il est à noter qu'il est possible de chainer les connecteurs lorsqu'une adaptation nécessite plusieurs opérations fournies par plusieurs connecteurs y compris lorsqu'il est nécessaire de changer d'interface réseau (*Wifi -> Zigbee par exemple*)

Nous proposant pour assurer la cohérence des applications visent à vis des changements du contexte une plate-forme d'adaptations dynamiques, l'adaptation dynamique est le processus par lequel une application logicielle est modifiée afin de prendre en compte un changement, que ce soit au niveau de l'environnement ou de l'application elle-même, cette plate-forme surveille et contrôle l'exécution des applications multimédia afin de détecter les changements éventuelles du contexte. En cas d'un changement, la plate-forme cherche les solutions possibles et prend la décision adéquate pour l'adaptation de l'application au nouveau contexte. Ensuite, la plate-forme cherche et choisie les services d'adaptation nécessaires afin de les intégrer dans des connecteurs d'adaptation et les réassemblés avec les composants métiers de l'application.

## IV. PROCESSUS D'ADAPTATION DANS LA PLATEFORME CSC

Le processus d'adaptation est une séquence des étapes à suivre pour choisir les services d'adaptation nécessaires et constituer les composants d'adaptation (*connecteurs*) qui seront utilisés pour assurer la communication entre des composants hétérogènes.

Le terme service est peut-être l'un des termes les plus utilisés et les plus ambigüe dans l'industrie du logiciel [3]. Habituellement, les services sont définis comme des fonctionnalités fournies par un système logiciel pour d'autres ou pour un utilisateur humain [4]. Dans le contexte des SOA, les services sont fournis par des prestataires de service indépendants qui instancient un logiciel sur leurs ordinateurs et publient les services qu'il fournit en utilisant des mécanismes normalisés afin qu'ils puissent être découverts et liés dynamiquement aux composants qui en ont besoin. Un service est un comportement défini par contrat qui peut être réalisé et fourni par tout composant pour être utilisé par tout autre sur la base unique du contrat [5].

L'adaptation dynamique est le processus par lequel une application logicielle est modifiée afin de prendre en compte un changement que ce soit au niveau de l'environnement ou de l'application elle-même. Il s'agit d'un processus en six étapes. Il faut tout d'abord (1) observer l'environnement d'exécution, (2) décider de l'opportunité de l'adaptation et de la stratégie appropriée à la situation détectée puis (4) rechercher les services d'adaptation nécessaires et (3) planifier les actions à réaliser pour adopter la stratégie décidée, puis (5) sélectionner les services d'adaptation capables d'assurer l'adaptation demandée et enfin (6) réaliser les traitements décidés. La figure3 présente un schéma explicatif de ce processus.

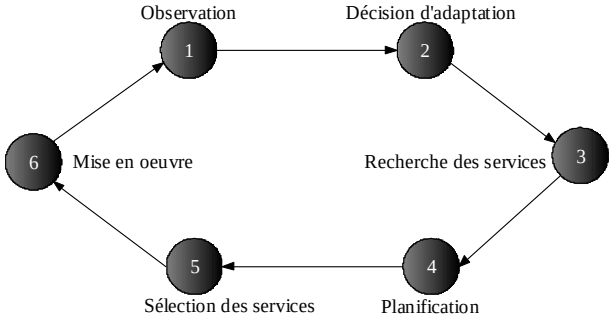



Figure 3. Processus d'adaptation dynamique

Pour chaque phase d'adaptation, plusieurs techniques sont utilisées. Un résumé de ces techniques est présenté dans le tableau suivant :

| Phases d'adaptations | Techniques utilisées |
|---|---|
| **Observation** | - *Capteur (détecteurs)*<br>- *Manuelle (observation)*<br>- *Monitoring* |
| **Décision** | - *Systèmes de règles*<br>- *Diagnostic basé sur un modèle*<br>- *Optimisation sous contraintes*<br>- *Modèles probabilistes*<br>- *Apprentissage automatique* |
| **Recherche de services** | - *Notion de découverte de service, UDDI (Universal Description Discovery and Integration)* |

| | |
|---|---|
| **Planification** | - *Théorie des graphes*<br>- *Réseau Pert*<br>- *Diagramme de Gantt*<br>- *Planification IA (Artificial Intelligence Planning)* |
| **Sélection des services** | - *Système de règles ;*<br>- *Programmation logique ;*<br>- *Recherche d'un chemin dans le graphe des adaptations.* |
| **Mise en œuvre** | - *Invocation du service SOAP (Simple Object Access Protocol) ;*<br>- *Programmation par aspect ;*<br>- *Programmation alternative.*<br>- *Programmation par composant* |

TABLE I. TECHNIQUES D'ADAPTATION

La phase de planification fournit un graphe d'adaptation constitue des services d'adaptation et des liaisons entre ces services et la phase de sélection fournit le chemin d'adaptation le plus adéquat.

Pour assurer l'auto-adaptation au niveau des applications, nous avons besoin de trois niveaux : la description, la supervision et l'adaptation.

La figure 4 présente une vue fonctionnelle de la plateforme de configuration et d'adaptation. Cette vue est composée d'un ensemble de descripteurs et de fonctions. Les fonctions sont distribuées sur trois niveaux : Description, Supervision et Adaptation. Les descripteurs de contexte et les manifestes des composants fournissent les exigences de l'application en matière de composants et d'adaptateurs afin de trouver la bonne configuration de l'application. La supervision permet de suivre les changements du contexte et de mettre à jour les exigences de l'application. Ces changements peuvent influer sur la configuration. L'adaptation assure une reconfiguration de l'application prenant en considération le problème d'hétérogénéité des composants.

## V. UNE APPROCHE D'ADAPTATION POUR L'INFORMATIQUE UBIQUITAIRE : CSC (COMPONENT SERVICE CONNECTOR)

La plateforme d'adaptation CSC est basée sur les concepts de composant, connecteur et service, les services sont utilisés par des connecteurs pour assurer des tâches d'adaptation, alors que les connecteurs sont utilisés pour assurer la communication et l'échange de données multimédia entre les composants.

Dans MMSA [1] nous avons présenté l'aspect architectural des applications multimédia et des mécanismes de vérification des configurations de ces derniers. La plateforme CSC s'intéresse à l'aspect comportemental et dynamique des mécanismes d'adaptation. Nous y aborderons les processus d'adaptation et d'auto-adaptation des applications multimédia et les mécanismes de choix et d'intégration de services d'adaptation.

Dans la plateforme CSC, nous cherchons à séparer la préoccupation d'adaptation des composants et de l'auto-adaptation des services d'adaptation des préoccupations fonctionnelles des applications, ce qui donne la possibilité de déléguer la complexité supplémentaire liée à l'adaptation et l'auto-adaptation à la plateforme. Le procédé d'adaptation s'applique sur le modèle de composant produit par MMSA (*Meta-model Multimedia Software Architecture*) qui permet de détecter les points d'hétérogénéités entre composant.

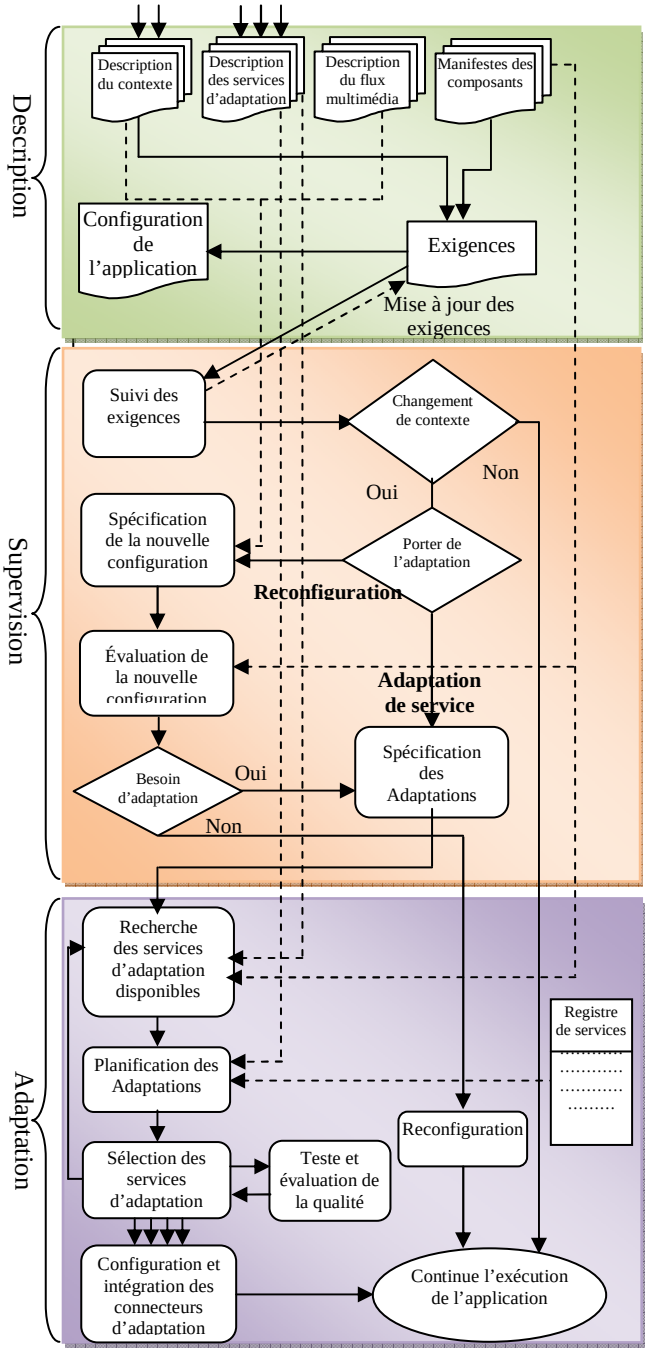



Figure 4. Processus de configuration/reconfiguration

Pour assurer l'adaptation des interactions entre composants en matière de flux de données, nous proposons une plateforme avec une architecture à trois couches (figure 5). Chaque couche assure une tâche dédiée, à savoir :

*a)* ***Couche configuration*** *: elle assure la reconfiguration dynamique de l'application, la détection des problèmes d'hétérogénéités inter-composants et des changements de contexte ;*

*b)* ***Couche d'adaptation :*** *elle assure la planification, la négociation et le choix des services d'adaptation ;*

*c)* ***Couche application*** *: elle est chargée de l'exécution des applications. Elle contient tous les éléments nécessaires à l'exécution des applications tels que les composants, les services ou les relations entre les services et les connecteurs. La découverte dynamique de changement du contexte, la supervision de l'exécution de l'application et la détection de violation des contrats sont des tâches assurées par le gestionnaire de QdS.*

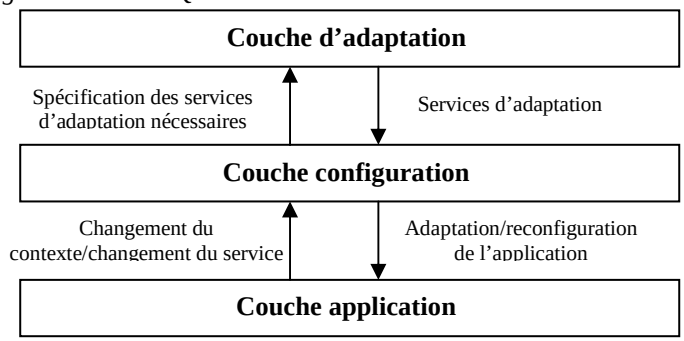



Figure 5. Différentes couches de la plateforme CSC

Deux facteurs intéressant doivent être pris en compte lors de la mise en œuvre d'une telle plate-forme, la portée de l'adaptation et son lieu d'exécution. Les adaptations peuvent être des changements d'interfaces (*aspect non-fonctionnel*), de services ou de composants (*aspect fonctionnel*). Au mécanisme de connecteur d'adaptation et de paramètres de service proposé dans MMSA, nous ajoutons la manipulation et le changement des paramètres de service qui permet d'influer sur les données produites par les services d'adaptation. Pour l'exécution des adaptations, il existe plusieurs approches : *basée client, basée serveur, Proxy et hybride.* Notre choix s'est porté sur l'approche hybride qui donne plus de flexibilité pour choisir l'endroit d'exécution des adaptations. C'est pourquoi nous avons choisi que l'exécution des connecteurs d'adaptation puisse avoir lieu au niveau client, serveur, ou intermédiaire, selon les capacités du client et le contexte de l'application.

### *A. Introduction à la plateforme CSC*

La gestion de la qualité de service représente une tâche importante dans la plateforme CSC. Pour cela, deux gestionnaires de QdS ont été proposés afin d'assurer la qualité de service au moment de la configuration de l'application et la qualité de service pendant l'exécution de l'application et des services d'adaptation. Le premier se situe au niveau de la plate-forme pour assurer la surveillance de la QdS en cours d'exécution de l'application. Il apporte les modifications nécessaires à l'adaptation de l'application au nouveau contexte. Le second se situe au niveau de chaque connecteur d'adaptation. Il gère la qualité des services d'adaptation à l'intérieur du connecteur par la modification des paramètres des services. Ce gestionnaire est en relation avec le gestionnaire de QdS de la plate-forme pour demander des changements de services d'adaptation en cas de nécessité.

La plateforme d'adaptation est chargée de la définition des architectures des applications basées composants et d'assurer leur adaptation au changement du contexte. Elle dispose d'un mécanisme de supervision pour assurer le suivi et le pilotage des applications. La plus part des applications intègrent : différents types de média, qu'ils soient discrets (*texte, images*) ou continus (*vidéo, audio*), des appareils mobiles avec des capacités diverses et des utilisateurs qui se déplacent dans des environnements informatiques ubiquitaires, ce qui impliquant des difficultés supplémentaires liées à l'auto-adaptation.

La plateforme CSC assure l'adaptation en vérifiant la cohérence des configurations des applications basées composants, pour ce faire elle propose un système d'adaptation orienté services afin d'augmenter la flexibilité en matière de recherche de services d'adaptation. L'exécution et le suivi des services d'adaptation sont assurés par des connecteurs d'adaptation.

Pour mieux comprendre l'objectif de la plate-forme CSC prenant l'exemple suivant :

Un utilisateur dispose d'un PC et une connexion sans fil (figure 6). Il veut suivre un match de football depuis un site internet qui diffuse le match en format *.rm (Real Player)* auquel il a un abonnement. Après 25 minutes de jeu, il a reçu un appel téléphonique pour rejoindre un ami à l'aéroport. Il est obligé de quitter la maison, mais veut quand même suivre le match. Pour cela il dispose d'un PDA avec une connexion sans fil 3G. En restituant la connexion, l'application de diffusion détecte ses nouveaux paramètres, elle trouve que le PDA n'a pas de lecteur « real media » et ne peut pas recevoir une diffusion dans le même format que sur le PC à cause des caractéristiques matérielles (*taille d'écran, contrainte d'énergie*) et de la qualité de la connexion (*bande passante, type de connexion*). Pour maintenir la connexion, l'application a besoin d'adapter la vidéo du match afin de répondre au nouveau besoin. On doit donc chercher un service d'adaptation qui transcode le format de la vidéo de *.rm* en *.MP4* qui lui est supporté par le PDA et un autre service qui diminue sa résolution afin de réduire encore plus la bande passante pour être compatible avec la 3G. Après la recherche, la sélection et l'intégration des services d'adaptation l'utilisateur peut regarder le match pendant son voyage à l'aéroport.

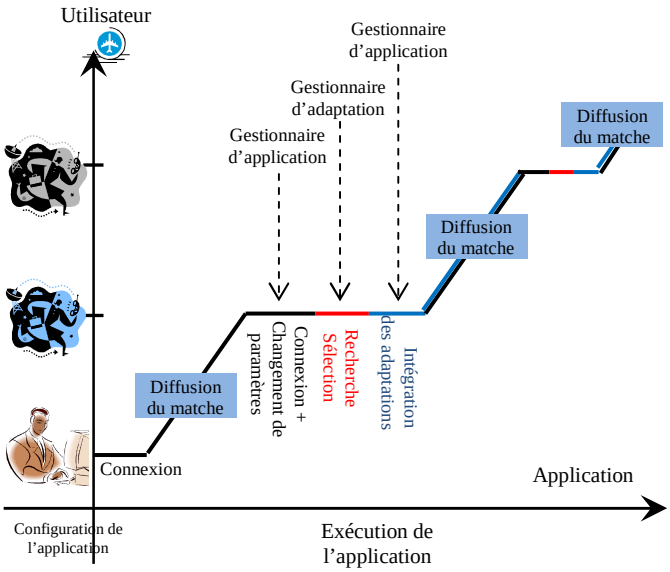



Figure 6. Scénarios de connexion de l'utilisateur

A la fin de la première période, la batterie du PDA est très faible, il décide de suivre le match sur son téléphone portable qui, pour des raisons de bande passante (*il n'a pas d'abonnement 3G*) ne peut recevoir que le son. Il s'agit là d'un scénario comparable dans lequel on cherche un service de transmodage (*changement de type de média*) de vidéo vers du son. Ainsi l'utilisateur pourra suivre le match jusqu'à la fin.

### *B. L'infrastructure de la plateforme CSC*

L'idée de base de CSC (figure 7) est d'utiliser les services fournis par les composants disponibles dans la plateforme et les services disponibles pour résoudre le problème d'hétérogénéité entre les composants d'une application. Il s'agit ici de services d'adaptation des données multimédia afin d'assurer une bonne configuration de l'application et une meilleure interopérabilité des composants. Le mécanisme de connecteur permet de modifier les paramètres des services d'adaptation, lors d'un changement de qualité ou de contexte.

La couche de configuration encapsule une bibliothèque de composants (F1) et de connecteurs (F2) utilisés pour la configuration des applications, une base de données pour sauvegarder les informations de contexte : utilisateurs (D1), logiciels (D2) et matériels (D3) et un gestionnaire d'applications qui représente le moteur de « raisonnement » de cette couche. Le gestionnaire d'applications (A) est composé d'un gestionnaire de contexte (A1), d'un gestionnaire de QdS (A2), d'un gestionnaire d'assemblage (A3) et d'un gestionnaire de configuration(A4). Le gestionnaire d'application utilise le modèle MMSA pour décrire l'architecture d'une application ce qui lui permet de détecter les besoins en adaptation entre les composants d'une configuration. Les points d'hétérogénéité sont détectés à partir d'une analyse des manifestes des composants (E1) et des éléments du contexte. Le gestionnaire de contexte (A1) est responsable de la mise à jour du contexte après une détection de changement annoncée par le gestionnaire de QdS (A2). Ensuite, le besoin d'adaptation est transféré à la couche d'adaptation sous forme d'une spécification d'adaptation. Cette couche encapsule les descripteurs de composants (E1), de connecteurs (E3) et de services (E2) d'adaptation ainsi qu'un gestionnaire d'adaptation (B). Le gestionnaire d'adaptation est composé d'un sélectionneur de service (B1), d'un planificateur d'adaptation (B2) et d'un négociateur de service (B3). Lorsqu'une demande d'adaptation arrive, le planificateur d'adaptation transforme la spécification d'adaptation en un graphe d'adaptation qui contient tous les chemins d'adaptation possibles suivant les services d'adaptation disponibles compte tenu des descriptions des connecteurs, des descripteurs des services et du registre de services (G1). Puis, le sélectionneur de services utilise le processus d'adaptation pour trouver le meilleur chemin d'adaptation en construisant un tableau des services d'adaptation afin de les classer par type et par qualité. Ce tableau servira par la suite pour changer les services d'adaptation en cas de nécessité. Puis, le sélectionneur de service demande au négociateur la négociation et l'établissement des contrats avec les fournisseurs de services (G2). Enfin le gestionnaire d'assemblages assure l'assemblage des composants et des connecteurs après intégration des services d'adaptation selon la configuration choisie.

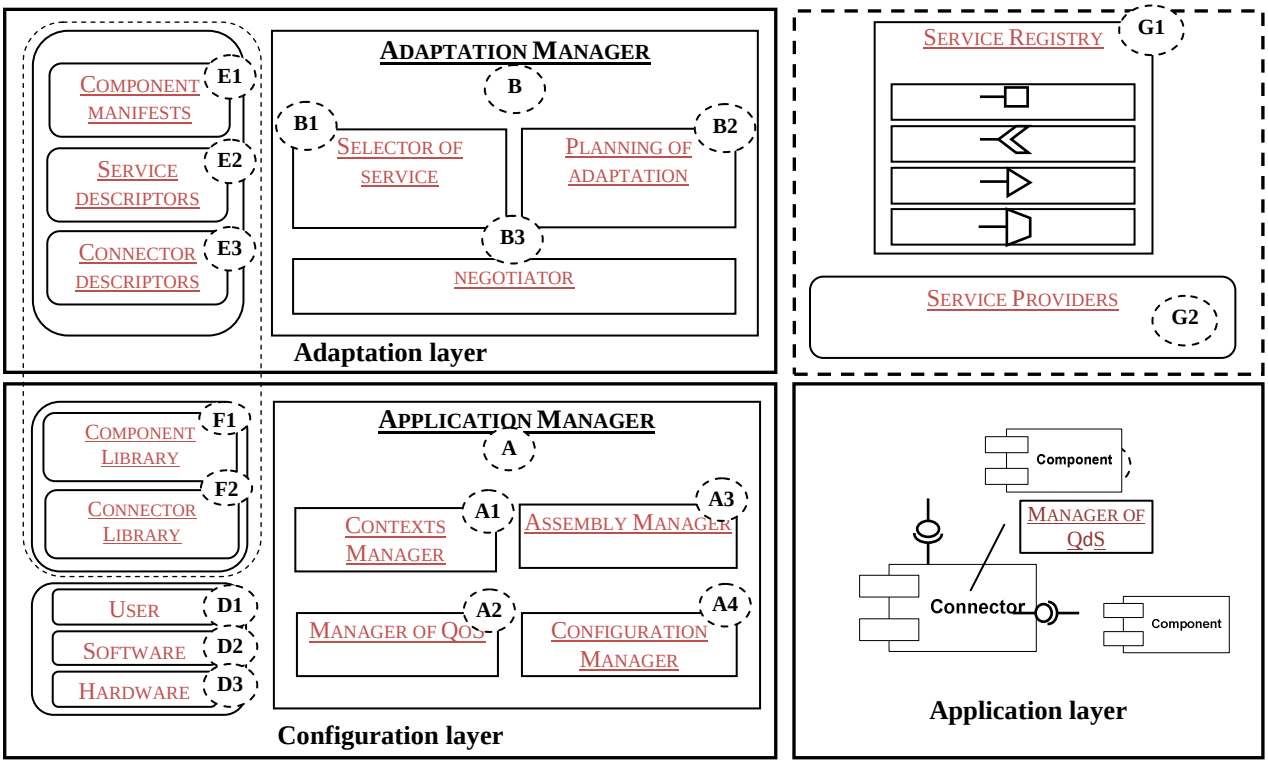



Figure 7. Plateforme CSC

## VI. DESCRIPTION DES COUCHES DE LA PLATEFORME CSC

La plateforme CSC (figure 7) est divisée en trois couches : la couche de configuration, la couche d'adaptation et la couche application. Le gestionnaire de configuration au niveau de la couche de configuration est chargé du choix des configurations et des composants adéquats, il est également chargé des opérations d'assemblage et de réassemblage des composants. Il utilise MMSA pour décrire l'architecture d'une application ce qui lui permet de connaître les besoins en adaptation des composants d'une application et de construire les connecteurs d'adaptation. Ces besoins seront par la suite transmis au gestionnaire d'adaptation. Le gestionnaire de QdS au niveau de la couche de configuration assure la résolution des conflits entre les configurations possibles et l'installation des services d'adaptation, ainsi que le suivi des applications et assurant le bon déroulement de l'application en vérifiant les changements du contexte. Le gestionnaire de contexte gère les changements de profils.

Le gestionnaire d'adaptation au niveau de la couche d'adaptation assure le choix des services d'adaptation. Cette tâche est assurée par trois services : le planificateur d'adaptation, le sélectionneur et le négociateur. Le premier fourni le plan d'adaptation pour chaque adaptation demandée, le deuxième assure la sélection des services d'adaptation, tandis que le dernier assure la négociation et l'établissement des contrats avec les fournisseurs de services.

### *A. Couche Application*

L'adaptation nécessite la détection des changements de contexte, mais aussi le choix d'une configuration de l'application qui maintienne une qualité satisfaisante dans le nouveau contexte. Il est donc nécessaire de découvrir dynamiquement les services d'adaptation dès qu'ils sont utiles ainsi que leurs disparition afin d'assurer leur remplacement. Ce travail est assuré par la couche de configuration.

Un gestionnaire de QdS existe au niveau de chaque connecteur d'adaptation, il informe le gestionnaire de QdS de la couche de configuration de toute nécessité de changement de service d'adaptation en raison d'une indisponibilité ou d'une insuffisance de qualité.

### *B. Couche Configuration*

Cette couche est composée d'un gestionnaire d'applications, de données qui décrivent le contexte (utilisateurs, logiciels, matériels) et d'une bibliothèque (composants et connecteurs).

Le gestionnaire d'applications est composé de quatre gestionnaires :

- Le gestionnaire de configurations fournit toutes les configurations possibles pour une application, ainsi, il est capable de détecter les incompatibilités entre les composants d'une configuration (hétérogénéité au niveau des flux de données) en analysant les manifestes des composants qui contiennent des détails sur les Entrées/Sorties des composants. Un manifeste permet de décrire les composants selon un modèle abstrait (sans détail d'implémentation). Il permet de séparer la description abstraite de la fonctionnalité offerte par un composant, ainsi que des détails concrets du composant tels que «comment» et «où» cette fonctionnalité est offerte et de décrire les données manipulées par les composants (type de données, format, contraintes temporelles – forte/faible -, etc.).
- Le gestionnaire de contextes assure la mise à jour du contexte lors de changement supervisé par le gestionnaire de QdS. Il fournit au gestionnaire de configurations les informations nécessaires sur l'environnement (utilisateur, logiciels, matériels). Par exemple pour une application Web : le type et la version du navigateur, le terminal de navigation, les préférences et caractéristiques physiques de l'utilisateur, etc. afin de mieux choisir la configuration et les composants.
- Le gestionnaire de QdS assure le contrôle et le suivi des applications en vérifiant les éventuels changements de contexte qui influencent le bon déroulement de l'application. Il coopère avec les gestionnaires de QdS au niveau de chaque connecteur d'adaptation.
- Le gestionnaire d'assemblages assure l'assemblage /réassemblage des composants et des connecteurs.

Après la détection d'un changement de contexte, la plateforme procède au départ à un changement de service d'adaptation afin de répondre a ce changement, si le changement des services d'adaptation n'est pas suffisant et ne permet pas de répondre au nouveau contexte la plateforme procède un changement de quelques composants suivie par un réassemblage ou une reconfiguration de l'application selon la nécessité. Dans le cas d'une reconfiguration, il faut choisir les composants de la nouvelle configuration et puis vérifier la cohérence de la configuration en utilisant l'approche MMSA [1]. Ensuite elle fait appel à la couche d'adaptation pour disposer des services nécessaires et les intégrer dans les connecteurs (figure 8).

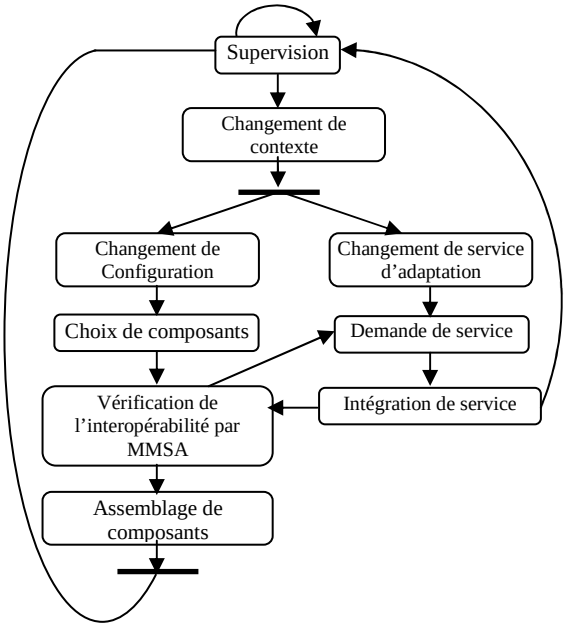



Figure 8. Diagramme d'activité de la plateforme de configuration

*C. Couche d'Adaptation*

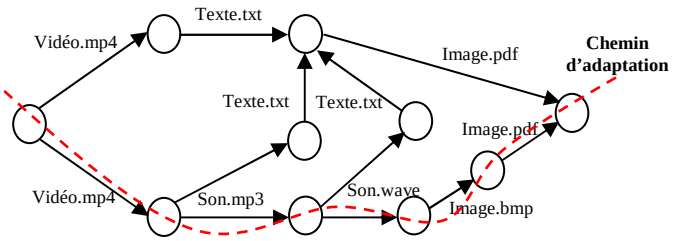
Cette couche est composée d'un gestionnaire d'adaptation et d'un ensemble de descripteurs de services, composants et connecteurs.

Le gestionnaire d'adaptations comporte trois composants :

- Le planificateur d'adaptation spécifie les adaptations demandées en matière de services d'adaptation. Ensuite, il consulte le registre de services à la recherche des services d'adaptation disponibles. Enfin, il construit un graphe d'adaptation dont les sommets représentent les services d'adaptation et les arcs les liens entre ces services.

**Exemple** : l'adaptation d'une vidéo d'une présentation orale dans une conférence en un fichier PDF.

Cette adaptation passe par plusieurs étapes : extraction du son, transmodage du son vers du texte, et transmodage du texte vers une image au format PDF. Ce qui fait en tout, trois services d'adaptation. En consultant le registre de service, nous pouvons trouver le graphe suivant :




- Le sélectionneur de services assure la sélection du meilleur chemin d'adaptation à partir du graphe dans le cas où on trouve plusieurs services qui font la même chose ou plusieurs chemins fournissant l'adaptation demandée. Pour cela, il utilise la théorie du graphe pour trouver le meilleur chemin d'adaptation (figure 9).

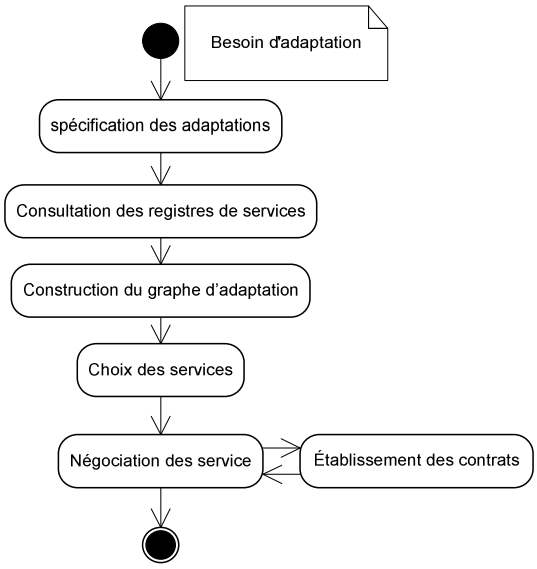



Figure 9. Diagramme d'activité de la plateforme d'adaptation

- Le concepteur affecte des poids aux critères de qualité (par exemple : résolution, taux de compression, etc.) selon leurs nécessités. Ensuite, il calcule la QoS de chaque service dans le graphe et puis la QoS de chaque chemin d'adaptation. La comparaison des QoS des chemins d'adaptation permet de choisir le meilleur chemin [22].

Il existe deux types de qualité de service : statique et dynamique. La gestion de la QdS statique est assurée par un processus de choix entre plusieurs services fournissant différentes qualités, tandis que la gestion de la QdS dynamique est assurée par le connecteur d'adaptation qui manipule les paramètres du service d'adaptation afin de satisfaire au contexte d'exécution.

- Le négociateur de services négocie avec les fournisseurs des services choisis et établit les contrats de services en utilisant le protocole SLA (Service Level Agreement).

Un concept de base pour les architectures orientées service est le contrat de service standardisé [6] qui est utilisé pour exprimer les services. Les propriétés de QdS sont généralement négociées entre le fournisseur et le consommateur de service, et sont décrites dans le cadre d'un contrat comme un SLA (Service Level Agreement). Un niveau de service est utilisé pour décrire la performance attendue (par exemple le temps de réponse et la disponibilité) et des propriétés telles que la facturation, les conditions de résiliation et les pénalités en cas de violation du SLA [7]. Dans notre système on ne s'intéresse qu'aux performances attendues.

**Exemple :**

Les contrats de niveau de service peuvent contenir de multiples mesures de performances (du service) correspondant aux objectifs du niveau de service. Prenons, par exemple, l'adaptation des images, les paramètres que l'on mesure dans ce cas sont généralement :

- TA : (Temps d'Adaptation) : temps moyen d'exécution du service d'adaptation.
- TT : (Temps de Transfert) : temps moyen de transfert d'une image adaptée.
- QA : (Qualité d'adaptation) : pourcentage de qualité de la sortie par rapport à l'entrée.

Un SLA peut être créé après avoir sélectionné un niveau de service fixé parmi plusieurs offres prédéfinies ou, dans des cas plus complexes, après une personnalisation via un processus de négociation. Un SLA peut être valable pour une période limitée (*adaptation d'un ensemble d'images par exemple*) ou être résilié de manière explicite (*fin d'une diffusion en direct par exemple*). Au cours du SLA, le fournisseur surveille le service et adapte ses ressources pour éviter des violations de SLA.

Le Gestionnaire de négociation est responsable de la négociation et de la réalisation des contrats SLA avec les fournisseurs choisis par le Gestionnaire d'adaptation. Après établissement des contrats de services, le gestionnaire de QdS du connecteur d'adaptation prend le contrôle et la surveillance des services d'adaptation. Si le service d'adaptation n'arrive pas à satisfaire les besoins des composants liés par ce connecteur d'adaptation, ce dernier demande le changement de

ce service auprès du gestionnaire de QdS de la couche de configuration.

## VII. DISCUSSION

Les adaptations des applications basées composants se réfèrent à la capacité d'un système à s'adapter aux besoins changeants des utilisateurs et au contexte en exploitant les connaissances sur sa configuration et les caractéristiques de la QdS de ses composants constitutifs. L'adaptation basée planification [8, 9, 10, 11, 12] est l'une des approches d'adaptation des applications basées composants. Dans MUSIC [13], cette connaissance est fournie sous la forme d'un modèle dirigé par la QdS qui décrit la composition abstraite, les dimensions pertinentes de la QdS et comment elles sont affectées lorsqu'il existe des variantes de configuration des composants. Ce modèle est exploité par le middleware d'adaptation pour sélectionner, connecter et déployer une configuration de composants fournissant la meilleure utilité. L'utilité est mesurée par le degré d'accomplissement des préférences utilisateur tout en optimisant l'utilisation des ressources du dispositif [14, 9].

Adaptive Service Grids (ASG) [15] et VieDAME [16] sont des initiatives permettant les compositions dynamiques et les attachements de services pour l'adaptation. En particulier, ASG propose un cycle de vie adapté à la livraison de services d'adaptation. Il est composé de trois sous-cycles: la planification, l'attachement de la spécification sémantique au service concret et l'incorporation. Le point d'entrée du cycle de livraison est une demande de services sémantiques qui consiste en une description de ce qui sera réalisé et non du service concret qui doit être exécuté. VieDAME propose un système de contrôle qui observe l'efficacité des processus BPEL (Business Process Execution Language) et effectue automatiquement le remplacement de service en cas de dégradation des performances. Comparé à notre approche, ASG et VieDAME sont basés seulement sur la planification à la demande de compositions de services qui concerne les propriétés définies dans la demande de service sémantique. Ainsi, les deux approches ne garantissent pas une configuration cohérente des composants et des services alors que notre plateforme l'assure pour des applications ubiquitaires basées composants tout en séparant les préoccupations de l'application de celles de l'adaptation assurée par les connecteurs d'adaptation.

Menasce et Dubey [17] proposent une approche de QdS en SOA. Les clients demandent des services au broker de QdS qui sélectionne un fournisseur de services qui maximise la fonction d'utilité du client en rendant compte dès contraintes de coût. L'approche suppose que les fournisseurs de services s'inscrivent auprès du broker en fournissant des demandes pour chacune des ressources utilisées par les services fournis ainsi que les fonctions de coût pour chaque service. Le broker de QdS emploie des modèles analytiques pour prévoir les valeurs de QdS des divers services qui pourraient être choisis dans des conditions variables de travail. Cette approche est intéressante du point de vue client et fournisseur. Tandis que le client est déchargé d'accomplir la découverte et la négociation de service, le fournisseur détermine le support de la gestion de QdS. Cette approche exige que le dispositif client permette l'accès au broker, ce qui pourrait ne pas être possible dans des environnements ubiquitaire. Notre approche diffère en ce que l'on considère les propriétés proposées comme solutions pour déterminer la meilleure configuration de l'application et permettre aux connecteurs de s'adapter aux besoins des composants.

CARISMA est un middleware pair-à-pair mobile exploitant le principe de réflexion pour supporter la construction des applications adaptables sensibles au contexte [18]. Les services et les politiques d'adaptation sont installés et désinstallés à la volée. CARISMA peut déclencher automatiquement l'adaptation des applications déployées lors de la détection des changements de contexte. CARISMA utilise des fonctions de service pour choisir les profils d'application qui sont utilisés afin de déterminer l'action appropriée à un événement particulier de contexte. S'il existe des profils d'application en conflit, il emploie une procédure d'enchère pour les résoudre. Contrairement à notre approche, CARISMA ne traite pas la découverte de services à distance qui peuvent déclencher des reconfigurations d'application et ne permet pas la recherche des nouvelles configurations possibles.

Les deux modèles conceptuels SeCSE (http://www.secse-project.eu/) et PLASTIC (http://www.ist-plastic.org/) se focalisent sur les systèmes orientés services. Le modèle PLASTIC est une extension du modèle SeCSE qui introduit de nouveaux concepts, comme le contexte, la localisation et le niveau de crédibilité du service. Notre approche et le modèle PLASTIC partagent l'approche de type SOA et le développement de logiciels basés composants. Toutefois, le modèle conceptuel de notre approche est centré composant tandis que celui de PLASTIC est centré service.

Le modèle MUSIC décrit la composition abstraite comme un ensemble de rôles en collaboration avec des ports qui représentent des fonctionnalités fournies ou requises par les composants collaborateurs. Les propriétés et les fonctions prédictives associées aux ports définissent comment les propriétés de QdS et les besoins en ressources des composants sont influencés par les propriétés de QdS des composants dont ils dépendent. Ce middleware d'adaptation propose un typage de service afin de pouvoir changer de service en cas d'absence ou de changement de contexte. Notre plateforme propose un typage des données afin de pouvoir détecter et régler le problème d'hétérogénéité entre composants manipulant différents types de média. Le typage de services est utilisé dans notre approche pour classer les services d'adaptation selon l'adaptation fournie ce qui nous donne la possibilité de choisir les meilleurs services en termes de qualité et de changer le service d'adaptation en cas de disparition ou de baisse de qualité.

## VIII. CONCLUSION

L'apparition récente d'une grande variété de moyens de communication, accompagnée d'un accroissement important de l'information multimédia rend la transformation et l'adaptation de contenu nécessaires. Cette nécessité se justifie par une demande croissante d'accès et d'échange de l'information en tout lieu et sur des plates-formes très hétérogènes.

La plate-forme CSC est une plate-forme d'exécution et d'adaptation des applications multimédia basées composants, elle assure l'adaptation des applications au contexte, en proposant en premier lieu des connecteurs d'adaptation pour gérer les changements provoqués par ce changement du contexte et, en second lieu, une reconfiguration de l'application qui tient compte du nouveau contexte et qui utilise l'approche MMSA pour valider la nouvelle configuration.

L'un des avantages importants des services est la possibilité de les composer afin de construire des services plus complexes de niveau sémantique généralement plus élevé. Dans le processus de composition, l'étape de sélection des services permet de choisir les services concrets qui seront les entités participantes à la composition de services.

La solution proposée est une plate-forme d'adaptation collaborative et distribuée qui, à partir d'une abstraction des composants appelée manifeste, arrive à détecter les points d'incompatibilité entre deux composants voisins selon l'approche MMSA. Ensuite, elle construit des variantes de connecteurs à partir des services d'adaptation fournissent par l'environnement collaboratif. Ensuite, elle permet de choisir et d'intégrer ces variantes dans des connecteurs dits d'adaptation. Enfin, elle assure l'assemblage/réassemblage de composants et de connecteurs d'adaptation qui peuvent dynamiquement sélectionner et exécuter la bonne adaptation (gestion de QdS). En utilisant ce processus de conception et d'adaptation, notre système reste cohérent.

## RÉFÉRENCE


[1] M. Derdour, P. Roose, M. Dalmau, N. Ghoualmi-Zine, A. Alti - MMSA: Metamodel Multimedia Software Architecture - Advances In Multimedia, Hindawi Ed. - vol. 2010, Article ID 386035, 17 pages, 2010. doi:10.1155/2010/386035.

[2] M. Derdour, P. Roose, M. Dalmau, N. Ghoualmi-Zine - Typing of Adaptation Connectors in MMSA Approach Case Study: Sending MMS International Journal of Research and Reviews in Computer Science (IJRRCS) - pp. 39-49, Vol. 1, No. 4, 12/2010 - ISSN: 2079-2557.

[3] Baida, Z., et al.: A shared service terminology for online service provisioning. In: 6th Int. Conf. on Electronic commerce, 2004.

[4] Sassen, A., Macmillan, C.: The service engineering area: An overview of its current state and a vision of its future. European Commission. Network and Communication Technologies, Sof Technologies, 2005.

[5] Zoran Stojanovic , Ajantha Dahanayake, « Service-Oriented Software System Engineering: Challenges and Practices », IDEA Group, 2005, ISBN 1-59140-426-6.

[6] Erl, T.: Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall, Englewood Cliffs, 2006.

[7] Dan, A., Ludwig, H., Pacifici, G.: Web service differentiation with service level agreements. IBM White Paper (2003)

[8] Rouvoy, R., et al.: Composing Components and Services using a Planning-based Adaptation Middleware. In: Pautasso, C., Tanter, É. (eds.) SC 2008. LNCS, vol. 4954, pp.52–67. Springer, Heidelberg, 2008.

[9] Geihs, K., et al.: A comprehensive solution for application-level adaptation. Software: Practice and Experience, 2008.

[10] Brataas, G., et al.: Scalability of Decision Models for Dynamic Product Lines. In: Int. Work. on Dynamic Software Product Line, DSPL, 2007.

[11] Floch, J., et al.: Using Architecture Models for Runtime Adaptability. IEEE Software, 2006.

[12] Lundesgaard, S.A., et al.: Construction and Execution of Adaptable Applications Using an Aspect-Oriented and Model Driven Approach. In: Indulska, J., Raymond, K. (eds.) DAIS 2007. LNCS, vol. 4531, pp. 76–89. Springer, Heidelberg (2007)

[13] Romain Rouvoy, Paolo Barone, Yun Ding, Frank Eliassen, Svein Hallsteinsen, Jorge Lorenzo, Alessandro Mamelli, and Ulrich Scholz. MUSIC: Middleware Support for Self-Adaptation in Ubiquitous and Service-Oriented Environments. Springer-LNCS 5525, pp. 164–182, 2009.

[14] Mascolo, C., Capra, L., Emmerich, W.: Mobile Computing Middleware. In: Gregori, E., Anastasi, G., Basagni, S. (eds.) NETWORKING 2002. LNCS, vol. 2497, pp. 20–58. Springer, Heidelberg, 2002.

[15] Kuropka, D., Weske, M.: Implementing a Semantic Service Provision Platform — Concepts and Experiences. Wirtschaftsinformatik Journal (1), 16–24, 2008.

[16] Moser, O., Rosenberg, F., Dustdar, S.: Non-intrusive monitoring and service adaptation for WS-BPEL. In: 17th Int. Conf. on World Wide Web (WWW). ACM, New York, 2008.

[17] Menasce, D., Dubey, V.: Utility-based QoS Brokering in Service Oriented Architectures. In: Int. Conf. on Web Services (ICWS), 2007.

[18] Capra, L., Emmerich, W., Mascolo, C.: CARISMA: Context-Aware Reflective Middleware System for Mobile Applications. IEEE Trans. on Software Engineering, 2003.

[19] Romain Rouvoy, Paolo Barone, Yun Ding, Frank Eliassen, Svein Hallsteinsen, Jorge Lorenzo, Alessandro Mamelli, and Ulrich Scholz. MUSIC: Middleware Support for Self-Adaptation in Ubiquitous and Service-Oriented Environments. Springer-LNCS 5525, pp. 164–182, 2009.

[20] G. Grondin, N. Bouraqadi, and L. Vercouter. MaDcAr: an Abstract Model for Dynamic and Automatic (Re-) Assembling of Component-Based Applications. In Proceedings of the 9th International SIGSOFT Symposium on Component-Based Software Engineering (CBSE 2006), LNCS 4063, pages 360-367, June 2006, Västerås, Sweden. Springer.

[21] Luc Fabresse, Christophe Dony, and Marianne Huchard. Foundations of a Simple and Unified Component-Oriented Language. Journal of Computer Languages, Systems & Structures, editor Elsevier, Volume 34/2-3 (July-October 2008), p. 130-149.

[22] Makhlouf Derdour, Nacira Ghoualmi-Zine, Philippe Roose, Marc Dalmau - Toward a dynamic system for the adaptation multimedia fluxes in the P2P architectures - FINA, helded in AINA-09, ISSN : 978-0-7695-3639-2/09. DOI 10.1109/WAINA.2009.28, pp 67-72.